Course Outlines

1st Semester

ANALYSIS III (FOR IC)

Summary
The course studies the fundamental concepts of vector analysis and Fourier analysis with a view to using them to solve multidisciplinary scientific engineering problems.

Content
Vector analysis:
Gradient, rotational, divergence and Laplacian operators. Curvilinear integrals and surface integrals. Vector fields and potentials. Green's, divergence and Stokes' theorems.
Fourier analysis:
Fourier series. Parceval identity. Fourier transforms. Plancherel's identity. Uses and applications.

Learning Prerequisites
Required courses
Analyse I, Analyse II, Algèbre linéaire.

SOFTWARE CONSTRUCTION

Summary

Learn how to design and implement reliable, maintainable, and efficient software using a mix of programming skills (declarative style, higher-order functions, inductive types, parallelism) and fundamental software construction concepts (reusability, abstraction, encapsulation, composition, proofs)

Content

Functional programming:

Functional programming paradigm
Recursion and tail-recursion
Evaluation strategies, lazy evaluation, substitution model
Modularity, data abstraction, representation independence
Subtyping, inheritance, type classes
Polymorphism, variance
Structural induction
Stateless parallelism, map-reduce, associative operations
Effects: state, exceptions
Documentation, tests, specification
Interpreters and program semantics
Software engineering:

Specifications: Documentation, requirements, properties
Verification: Debugging, tests, monitoring, property-based testing, proofs
Modularity: Code evolution and refactoring
Collaboration: Version control, changelogs

Learning Prerequisites

Required courses
Any previous programming course

Recommended courses
CS-107 Introduction à la programmation

CS-108 Pratique de la programmation orientée-objet

Important concepts to start the course
Loops, conditionals, variable and type declarations, computing mathematical expressions

Learning Outcomes

Implement reliable, efficient, and maintainable software
Identify data types and operations that lead to computational solutions
Argue that an solution is correct
Transform a program to change its behavior in a desirable way
Design and implement data-parallel software using parallel collections

Teaching methods

Ex cathedra (live lectures)

Recorded videos
Exercise and lab sessions
Online discussions
Expected student activities
Attending lectures
Watching and understanding recorded videos
Solving exercises individually or in groups
Completing individual graded programming assignments (labs / mini-projects)
Completing midterm and end-of-semester exams
Assessment methods
30% Midterm exam during the semester
40% Final exam during the exam session
30% Programming assignments (labs)

PROBABILITY AND STATISTICS (FOR IC)

Summary
A basic course in probability and statistics

Content
Revision of basic set theory and combinatorics.

Elementary probability: random experiment; probability space; conditional probability; independence.

Random variables: basic notions; density and mass functions; examples including Bernoulli, binomial, geometric, Poisson, uniform, normal; mean, variance, correlation and covariance; moment-generating function; joint distributions, conditional and marginal distributions; transformations.

Many random variables: notions of convergence; laws of large numbers; central limit theorem; delta method; applications.

Statistical inference: different types of estimator and their properties and comparison; confidence intervals; hypothesis testing; likelihood inference and statistical modelling; Bayesian inference and prediction; examples.

Learning Prerequisites
Required courses
Analyse I, II

Algèbre linéaire

Teaching methods
Ex cathedra lectures, exercises and problems

Assessment methods
Written exam

COMPUTER ARCHITECTURE

Content
Complex digital systems in Verilog
Basic components of a computer
Instruction Set Architectures (RISC-V)
Memory Hierarchy
IOs and Exceptions
Instruction Level Parallelism
Multiprocessors and Cache Coherence
Learning Prerequisites
Required courses
CS-173 Fundamentals of digital systems

Recommended courses
None

Important concepts to start the course
Digital logic (combinational and sequential circuits, FSMs)
Basic notions of processors and assembly
Learning Outcomes
By the end of the course, the student must be able to:

Structure nontrivial assembly language programs
Add interrupt handling logic in a processor and write simple exception handlers in assembler
Understand the design principles of a modern memory hierarchy
Understand the interaction mechanisms of system software with hardware
Design pipelined digital circuits at Register Transfer Level
Optimize the performance of a processor pipeline by reordering instructions
Teaching methods
Ex cathedra
Exercises
Projects
Expected student activities
Attending the course and exercise/lab sessions
Completing the exercises and lab assignments
Participating in the discussion on the forum
Assessment methods
Graded lab assignments
Midterm exam
Written exam

ALGEBRA

Summary
This is an introduction to modern algebra: groups, rings and fields.

Content
Integer numbers, Bezout's theorem. Groups, dihedral and symmetric groups. Group homomorphisms. Classification of finite abelian groups. Rings, ideals. Polynomial rings. Integral domains and Euclidean domains. Finite fields.

Keywords
Group, homomorphism, subgroup, normal subgroup, quotient group, cyclic group, symmetric group, order of the group, order of an element in the group, finite abelian groups. Ring, ideal, principal ideal, maximal ideal, principal ideal domain, Euler's totient function, field, finite field, characteristic of a field.

Learning Prerequisites
Required courses
Linear algebra

Recommended courses
Linear Algebra I, Analyse I, Analyse II

Learning Outcomes
By the end of the course, the student must be able to:

Apply concepts and ideas of the course
Reason rigorously using the notions of the course
Choose an appropriate method to solve problems
Identify the concepts relevant to each problem
Apply concepts to solve problems similar to the examples shown in the course and in problem sets
Solve new problems using the ideas of the course
Implement appropriate methods to investigate the structure of a given group, ring or field, and study their properties
Detect properties of algebraic objects
Analyze finite groups
Formulate structure of a finite abelian group in terms of cyclic groups
Analyze structure of a ring, in particular polynomial rings
Teaching methods
Lectures and exercise sessions

Assessment methods
Written homework assignment (10% of the grade)

Written exam (90 % of the grade)

ELECTRONICS I

Summary
Discover the world of electronics from the fundamental laws of discrete, linear and nonlinear components. Circuits obtained with component assemblies require numerous modeling and analysis techniques as well as simulator-based verifications

Content
Course

Linear passive components
Linear circuit solving techniques
Diodes
Introduction to Transistors
Techniques for modeling nonlinear components
Electronic simulation
Exercises
The student will apply the many methods seen in class to solve practical exercises that can be verified with the simulation.

Keywords
Passive components, active components, linear components, nonlinear components, diodes, transistors, modeling, simulation, Kirchhoff's laws, Thévenin-Norton, Superposition, complex impedances, transfer functions, Bode, amplification concept.

Required skills
Mandatory Prerequisites
Analysis course: first and second order differential equations, complex numbers, solving a system of linear equations.

Indicative prerequisites
Basic electricity: electrostatic, electrokinetic.

Learning outcomes
At the end of this course, the student should be able to:

Analyze complex circuits
Model nonlinear components
Modeling complex circuits
Reasoning from observation methods
Drawing temporal and frequency behaviors
Interpreting signals of various kinds
Use the right resolution methods
Compétences transversales
Utiliser les outils informatiques courants ainsi que ceux spécifiques à leur discipline.
Auto-évaluer son niveau de compétence acquise et planifier ses prochains objectifs d'apprentissage.
Assessment Methodology

Several quizzes spread over the semester and at least one written assignment.

NUMERICAL METHODS FOR VISUAL COMPUTING AND ML

Summary

Visual computing and machine learning are characterized by their reliance on numerical algorithms to process large amounts of information such as images, shapes, and 3D volumes. This course will familiarize students with a range of essential numerical tools to solve practical problems in this area.

Content

This course provides a first introduction to the field of numerical analysis with a strong focus on visual computing and machine learning applications. Using examples from computer graphics, deep neural networks, geometry processing, computer vision, and computational photography, students will gain hands-on experience with a range of essential numerical algorithms.

The course will begin with a review of floating point arithmetic and error propagation in numerical computations. Following this, we will study and experiment with several techniques that solve systems of linear and non-linear equations and perform dimensionality reduction. Since many interesting problems cannot be solved exactly, numerical optimization techniques constitute the second major topic of this course. We will take an extensive look at automatic differentiation, the mechanism underlying popular deep learning frameworks such as PyTorch and Tensorflow. The course concludes with a review of numerical methods that introduce randomness to solve problems that would otherwise be intractable.

Students will have the opportunity to gain practical experience with the discussed methods using programming assignments based on Scientific Python.

Keywords

Visual computing, machine learning, numerical linear algebra, numerical analysis, optimization, scientific computing

Important concepts to start the course

Students are expected to have good familiarity with at least one programming language (e.g. C/C++, Java, Scala, Python, R, Ruby...). The course itself will rely on Python, but this is straightforward to learn while taking the course.

During the first weeks of the semester, there will be tutorial sessions on using Python and Scientific Python.

Learning Outcomes

By the end of the course, the student must be able to:

Develop computer programs that use numerical linear algebra and analysis techniques to transform and visualize data.
Reason about ways of structuring numerical computations efficiently.
Analyze the numerical stability of programs built on top of floating point arithmetic.

Recognize numerical problems in visual computing applications and cast them into a form that can be solved or optimized.
Teaching methods
Lectures, interactive demos, theory and programming exercises

Expected student activities
Students are expected to study the provided reading material and actively participate in class and in exercise sessions. They will be given both theoretical exercises and a set of hands-on programming assignments.

Assessment methods
1. Continuous assessment during the semester via project assignments (35%)

2. Final exam (65%)

RESPONSIBLE SOFTWARE

Summary

Software's growing importance increases engineers' responsibility to integrate ethical concerns in the design and development process. This course teaches students concrete strategies for responsible software engineering, focusing on identifying ethical issues and mitigating risks to minimize harms.

Content

The course combines a) knowledge about a range of ethical challenges related to software, and b) pragmatic tools/strategies that students can use in practice to identify and work on ethical risks in the domain of software.

The course will use cases inspired or based on real-world examples to review a range of ethical challenges related to software, including the following themes and questions:

Safety: How to anticipate potential harmful impacts resulting from the normal and abnormal use of software at the scale of the individuals, groups or society? What types of mechanisms can be used to protect users from harmful impacts and what are their limits?
Fairness: What is bias in software? What are its sources and what types of impacts can it have? How to identify and mitigate fairness issues?
Sustainability: What influences the environmental impact of software in terms of energy consumption, CO2 emission and use of resources? How to estimate and limit this impact?
Transparency and Autonomy: What issues arise from a lack of transparency in software, and how can methods help users understand software functions and limits? How does software design affect user control and which factors contribute to user empowerment or disempowerment?
Through the course activities, students will get to learn and practice concrete strategies for a responsible approach to software design and development including:

User and stakeholder analysis strategies
Strategies for analyzing values
Impact anticipation strategies
Risk assessment strategies
Decision-making strategies
Note: Other concerns, such as security and privacy, are not directly addressed in this course since they are already covered in other courses in the curriculum. However they may be involved in the cases used in the course.

Learning Prerequisites

Required courses
Introduction à la programmation (CS-107)

Important concepts to start the course
Basics of imperative programming

Learning Outcomes

By the end of the course, the student must be able to:

Assess / Evaluate the level of responsibility of existing software using a set of ethical lenses
Identify ethical questions during software design and development
Integrate environmental and social concerns into the software design and development process
Take into consideration the perspectives of different stakeholders in a software project
Examine stakeholder values and identify value tensions in a software project
Investigate the potential benefits and potential harms of software for different stakeholders
Assess / Evaluate the ethical risks associated with software
Make design or development decisions taking ethical risks into account
Transversal skills
Take account of the social and human dimensions of the engineering profession.
Take responsibility for environmental impacts of her/ his actions and decisions.
Demonstrate the capacity for critical thinking
Demonstrate a capacity for creativity.
Use both general and domain specific IT resources and tools
Access and evaluate appropriate sources of information.
Teaching methods
The course is designed with a flipped classroom format, based on online resources from the "Responsible Software" MOOC, developed by the Center for Digital Education.

Each week begins with a problem representative of the topic before introducing the corresponding theoretical content, in the following typical sequence called "problem-solving before instruction":

In-class problem-solving on computer (analysis, design or development), with assistants;
Independent study of theoretical content (concepts, principles, strategies...) in the form of videos and/or reading material, guided by short activities such as quizzes or open-ended questions;
In-class application to one or more concrete case(s) in class with group activities.
Expected student activities
Complete the exercises
Study resources (videos and/or readings) and complete preparation activities (quizzes) prior to classroom sessions
Apply learning to in-class case studies
Assessment methods
Graded assignments (40%): 2 assignments (20% each) during the semester
Final exam (60%): 1 individual written test during the semester
Supervision

TECHNOLOGIES FOR DEMOCRATIC SOCIETY

Summary

This course will offer students a broad but hands-on introduction to technologies of human self-organization.

Content

The course will present students with a view of self-organization technologies set in a long-term historical perspective, extending from their roots in ancients principles of democracy and governance, up to recent high-tech innovation such as social networking, e-voting, blockchains, and delegative democracy. The course will cover the many fundamental organization challenges these technologies attempt to address, such as :

Coordination : do participants communicate in person, electronically, or by passing secret notes ?
Membership : who has the right to participate as a member or citizen ? Can membership be faked ?
Equity or fairness : how much power or weight does each participant have ? Can weight be hacked ?
Filtering : how to separate signal from noise, real expertise from appealing bluster ?
Scalability : does the self-organizing technology work for only 10 members, or 100? 1000 ? 1 M ? 1 B ?
Integrity : how does self-organizing technology prevent hacking or tampering by malicious parties ?
Self-determination : does the technology protect freedoms such as expression and association ?
Privacy : what acts of participation does the technology keep private, and what are considered public?
Representation : is participation direct or representative ? How are representatives chosen ?
Accountability : how are participants and/or representatives kept accountable for their actions ?
Transparency : does the technology allow participants to verify that it is operating correctly ? How ?
Incentives : how does the technology encourage or incentivize people to use it, for good or ill ?
Psychology : how does the technology interplay with the unique properties of the human mind ?

Learning Prerequisites

Important concepts to start the course
Basic computing and programming skills

Learning Outcomes

By the end of the course, the student must be able to:

Explore technologies available for societal self-organization
Expound key challenges and risks in using these technologies
Discuss social implications of digital communication and organization technologies

Teaching methods

The course will use readings, discussions, and exercises to lead students through an exploration of the vast number of different technological approaches to these challenges and issues, from extremely low-tech (e.g., picking representatives by drawing straws) to the latest experimental technologies. In different weeks the students will explore hands-on the architecture, design, practical use, and strenghts and weaknesses of different self-organization technologies, such as :

Public discussion forums such as UseNet, Twitter, and Reddit
Community self-organization systems such as Loomio
Peer review systems such as HotCRP
E-voting systems in use in around the world (especially the US and Switzerland)
Experimental participatory delegative democracy systems such as LiquidFeedback
Cryptocurrencies and smart contract systems such as Bitcoin and Ethereum
The course work will involve a substantial amount of reading background materials, both technical and non-technical and from a variety of disciplines including computer science, social science, political science, and law. The lectures will be heavily discussion-oriented, covering both the background readings and hands-on exercises in addition to material presented in the lectures.

Expected student activities
The course will encourage students to "learn by doing" through exercises with practical systems. Students will be required to use some of these systems in groups in "hands-on" self-organization exercises, to get firsthand comparative experience of how they work, and in what ways they succeed and fail.

Assessment methods
Students will be assessed through regular exercises and mini-quizzes, participation in "peer review" activities, a small project in the second half of the semester on which the students must report, and a written final exam. Grading will be based substantially on demonstrated active participation in the deliberative course exercises, in addition to learning and understanding of the course content itself.

ANALYTICAL MECHANICS (FOR SPH)

Summary
Presentation of the methods of analytical mechanics (Lagrange and Hamilton equations) and introduction to the notions of normal modes and stability.

Content
1. Reminders of Newtonian mechanics

2. Lagrange's equations- D'Alembert's principle.- Principle of least action.- Normal coordinates.

3. Hamilton's equations- Poisson's brackets.- Canonical transformations.- Hamilton-Jacobi method.

4. Introduction to dynamical systems- Notion of stability.- Normal modes.

Required skills
Indicative prerequisites
General Physics, Analysis, Linear Algebra

Learning outcomes
At the end of this course, the student should be able to:

Solving a mechanical problem
Transversal skills
Self-assess your level of acquired competence and plan your next learning objectives.
Teaching method
Ex cathedra and exercises in the room.

Assessment Methodology
Written exam

GENERAL PHYSICS: ELECTROMAGNETISM

Summary
The course deals with the concepts of electromagnetism, with the support of experiments. Topics covered include electrostatics, electric current and circuits, magnetostics, electromagnetic induction, Maxwell's equations, electromagnetic waves, and light as a wave.

Content
Electrostatics: Coulomb's law and electric field, Gauss's law, electric potential, capacitance and energy, electric fields in dielectrics.
Electric current and DC circuits.
Magnetostics: currents as the source of the magnetic induction field, the fundamental laws, the magnetic properties of matter.
Electromagnetic induction: electromotive force, induction law, inductors, magnetic energy, AC circuits.
Maxwell's equations: the displacement current and the equations in a vacuum (electromagnetic waves), Poynting vector and EM energy.
Light as a wave: propagation in a dielectric, reflection, refraction, standing waves, interference and diffraction.
Learning outcomes
At the end of this course, the student should be able to:

Synthesize the concepts illustrated in the course
Manipulate the equations given in the course
Solve problems (practical and theoretical) using the concepts given in the course
Choose or select to solve a problem
Inferring properties from fundamental laws
Teaching method
Ex cathedra with presentation of experiences.

Assessment Methodology
100% Written Exam

2nd Semester

ALGORITHMS I

Summary
The students learn the theory and practice of basic concepts and techniques in algorithms. The course covers mathematical induction, techniques for analyzing algorithms, elementary data structures, major algorithmic paradigms such as dynamic programming, sorting and searching, and graph algorithms.

Content
Mathematical Induction

Mathematical background, Euler's formula for trees.
Analysis of Algorithms

O-notation, time and space complexity, recurrence relations, probabilistic analysis.
Data structures

Arrays, linkes lists, trees, heaps, hashing, graphs.
Design of algorithms by induction

Divide-and-conquer algorithms, dynamic programming.
Greedy Algorithms

Spanning tree and shortest path algorithms.
Sorting and searching

merge sort, bucket sort, quicksort, heapsort, binary search.
Graphs algorithms and data structures

Graphs traversals, shortes path, spanning trees, transitive closures, decompositions, matching, network flows.
Keywords
Algoriths, data structures, efficiency, problem solving

Learning Outcomes
By the end of the course, the student must be able to:

Illustrate the execution of algorithms on example inputs
Describe basic data structures such as arrays, lists, stacks, queues, binary, search trees, heapas, and hash tables
Analyze algorithm efficiency
Compare alternative algorithms and data structures with respect to efficiency
Choose which algorithm or data structure to use in different scenarios
Use algorithms and data structures taught in the course on concrete problem instances
Design new algorithms and data structures bases on known methods
Prove the corrrectness of an algorithm

COMPUTER SYSTEMS

Language: English

Summary
This course will teach operating systems and networks in an integrated fashion,emphasising the fundamental concepts and techniques that make their interaction possible/practical. Core lectures will be coupled with C programming lectures and assignments for hands-on experience.

Content
We will communicate these concepts and techniques through examples at different layers of the stack and draw connections/parallels between different aspects of computing systems.

The role of the Operating System
File systems
Virtual memory
Efficient resource management
Networked applications
The Internet
Transport layer
Network layer
Link layer
Data-centers and cloud systems
Learning Prerequisites
Required courses
Important concepts to start the course
Basic Programming
Basic computer architecture
Learning Outcomes
By the end of the course, the student must be able to:

Manage key elements of operating systems and networks
Critique the design of an OS or network protocol
Design and implement C programs and network applications
Compare different OS functions and network protocols
Investigate simple OS and network attacks
Investigate the correctness of C programs through debugging
Teaching methods
Ex cathedra
Hands-on exercise sessions
Expected student activities
Participate in lectures and exercise sessions
Answer quizzes
Submit programming assignments
Take midterm and final exams
Assessment methods

Programming assignments (by groups of two)
Midterm and final exam
Quizzes

ANALYSIS IV

Summary
The course studies the fundamental concepts of complex analysis and Laplace analysis with a view to their use to solve multidisciplinary scientific engineering problems.

Content
Complex analysis
- Definitions and examples of complex functions.
- Holomorphic functions.
- Cauchy-Riemann equations.
- Complex integrals and Cauchy formulas.
- Laurent series.
- Residue theorem.

Laplace analysis
- Laplace transforms.
- Applications to ordinary differential equations.
- Applications to partial differential equations.

Learning Prerequisites
Required courses
Linear algebra, Analysis I, Analysis II, Analysis III

Assessment methods
Written exam

COMPUTER GRAPHICS

Summary
The students study and apply fundamental concepts and algorithms of computer graphics for rendering, geometry synthesis, and animation. They design and implement their own interactive graphics programs.

Content
This course provides an introduction to the field of Computer Graphics. We will cover elementary rendering algorithms such as rasterization and raytracing, examine mathematical concepts and algorithms for geometric modeling, and then study concepts and algorithms for computer animation. Students will experiment with modern graphics programming and build small interactive demos. Complemented by some theoretical exercises, these programming tasks lead to a graphics software project, where small teams of students design and implement a complete graphics application.

Keywords
Pixels and images, 2D and 3D transformations, perspective transformations and visibility, rasterization, interpolation and lighting, raytracing, shader programming, texture mapping, procedural modeling, curves and surfaces, polygonal meshes, particle systems

Learning Prerequisites
Required courses
Linear Algebra, Calculus

Learning Outcomes
By the end of the course, the student must be able to:

Explain and apply the fundamental mathematical concepts of computer-based image and geometry synthesis
Implement a basic rendering pipeline based on rasterization and raytracing
Design and implement geometry synthesis based on procedural modeling
Design and implement basic computer animation algorithms
Integrate individual components into a complete graphics application
Coordinate a team during a software project

Teaching methods
Lectures, interactive demos, theory and programming exercises, programming project, project tutoring

Expected student activities
The student are expected to study the provided reading material and actively participate in class. They should prepare and resolve the exercises, prepare and carry out the programming project. Exercises and project are done in groups of three students.

Assessment methods
Programming homeworks and group project
Final written examination

HUMAN COMPUTER INTERACTION
Remark: This course will be given for the last time in the spring of 2025

Summary
The discipline of Human-Computer Interaction (HCI) aims to systematically place the human factor in the design of interactive systems.

Content
Course Introduction

Introduction: Human-Computer Interaction & User Experience (UX)
Project phases

Gathering organizational needs and analyzing the competition: Voice of Business & Voice of the Market
Understand Target Users and Their Needs: Voice of the Customer
Specify target users and their usage contexts: Personas, User Journeys & UX Vision
Structure and name content: Information architecture
Design the Solution: Mock-up & Prototyping
Evaluate the solution: User testing
Thematic focus

Cognitive load
Data vizualisation
Web Accessibility
Interaction styles
Deceptive Patterns
Conclusions

Summary of key concepts & review of previous exams
Learning outcomes
At the end of this course, the student should be able to:

Detecting Interface Defects
Explain defects from the user's perspective
Suggest improvements to an interface
Carry out a usability test
Design an experience
Analyze a user's data
Recommend an interaction style for a given task
Plan the different phases of a user-centered design approach
Defining a UX problem
Identify design best practices
Determine the defects of an interface
Explain defects from the user's perspective
Propose recommendations for optimizations of an interface
Analyze qualitative and quantitative data
Create personas and user journeys

Design ergonomic interfaces in an iterative way
Conduct a user test
Take into consideration web accessibility criteria
Teaching method
Ex-cathedra courses including examples close to practice

Group project allowing students to put into practice the concepts of the course through the design of a solution focused on the needs and expectations of users

Expected work
Create an interactive prototype illustrating the main screens and user journeys of the system to be optimized

Assessment Methodology
Project (50%)

Written exam (50%)

INTRODUCTION TO MACHINE LEARNING

Summary

Machine learning and data analysis are becoming increasingly central in many sciences and applications. In this course, fundamental principles and methods of machine learning will be introduced, analyzed and practically implemented.

Content

Introduction: General concepts, data representation, basic optimization.
Linear methods: Linear regression, least-square classification, logistic regression, linear SVMs.
Nonlinear methods: Polynomial regression, kernel methods, K nearest neighbors
Deep learning: Multi-layer perceptron, CNNs.
Unsupervised learning: Dimensionality reduction, clustering.

Learning Prerequisites

Required courses
Linear Algebra

Important concepts to start the course
Basic linear algebra (matrix/vector multiplications, systems of linear equations, SVD)
Multivariate calculus (derivatives w.r.t. vector and matrix variables)
Basic programming skills (labs will use Python).

Learning Outcomes

By the end of the course, the student must be able to:

Define the following basic machine learning problems : regression, classification, clustering, dimensionality reduction
Explain the main differences between them
Derive the formulation of these machine learning models
Assess / Evaluate the main trade-offs such as overfitting, and computational cost vs accuracy
Implement machine learning methods on real-wolrd problems, and rigorously evaluate their performance using cross-validation

Teaching methods

Lectures
Pen-and-paper exercise sessions
Python lab with a mini project in groups of 3 students

Expected student activities

Attend lectures
Attend lab sessions
Work on the weekly theory and coding exercises

Assessment methods

Self-assessment via the solutions of the pen-and-paper exercises and coding labs
Two milestones for the mini-proejct (10% of the grade each)
Final exam (80% of the grade)

SIGNAL PROCESSING

Summary
Signal processing theory and applications: discrete and continuous time signals; Fourier analysis, DFT, DTFT, CTFT, FFT, STFT; linear time invariant systems; filter design and adaptive filtering; sampling; interpolation and quantization; image processing, data communication and control systems.

Content
Signal processing is an engineering discipline that studies how to analyze, modify, and transmit information using mathematical models, practical devices, and numerical algorithms.
The class will cover the following topics:

1. Basic discrete- and continuous-time signals and systems: signal classes and operations on signals, signals as vectors in a vector space
2. Fourier Analysis: properties of Fourier transforms, DFT, DTFT, CTFT; practical Fourier analysis (FFT, STFT)
3. LTI systems: properties and composition, convolution, application of Fourier analysis to LTI System, Laplace and z-Transforms.
4. Analog vs Digital: sampling, interpolation and quantization.
5. Applications: adaptive filtering; image processing, data communication and control systems.

Learning Prerequisites
Required courses
Linear Algebra, Programming (Python), Analysis II

Recommended courses
Analyse III (concurrently), Probability theory (concurrently)

Important concepts to start the course
Vectors and vector space, functions and sequences, infinite series

Learning Outcomes
By the end of the course, the student must be able to:

Identify signals and signal types
Describe properties of LTI systems
Analyze LTI systems by spectral analysis
Recognize signal processing problems
Apply the correct analysis tools to specific signals
Implement signal processing algorithms
Design digital filters
Interpret complex signal processing systems
Teaching methods
This course will weave together theoretical analysis in course lectures with practical hands-on labs using Python (via Jupyter notebooks) and more traditional exercise sessions.

Expected student activities
Study class material; complete weekly homework sets (with solutions discussed in subsequent exercise sessions) and participate in Python applied labs.

Assessment methods
The final grade will be almost fully determined by the final exam, with a small grade component based on compilation of weekly laboratory and homework assignments.

INTRODUCTION TO LIFE SCIENCES (FOR IC)

Summary

This course introduces the fundamentals of living organisms. As much as possible, the emphasis is placed on the contributions of Computer Science to the progress of the Life Sciences.

Content

The course covers most of the fundamental concepts of Life Sciences.

Topics developed because they interface with computer science include:

- sequence alignment, assembly of sequences in genome

- Distance matrix and inference of a phylogenetic tree

- Detection of transmembrane domains and subcellular localization signals in an amino acid sequence.

Keywords

Bioinformatics, genome, sequencing, evolution, intercellular communication

Learning outcomes

At the end of this course, the student should be able to:

Identify the main cellular structures and understand the methods used to observe them
Identify the informative segments of a genome
Apply algorithms to solve questions
Explain the process of gene expression
Analyze raw experimental data and draw sensible conclusions

Transversal skills

Access and evaluate appropriate sources of information.
Communicate effectively and be understood by people of different languages and cultures.

Expected work

In addition to active participation in classes and exercises, 4 hours of personal work are expected.

Assessment Methodology

Written exam during the summer session.

3rd Semester
COMPUTER SECURITY AND PRIVACY

Summary
This is an introductory course to computer security and privacy. Its goal is to provide students with means to reason about security and privacy problems, and provide them with tools to confront them.

Content
The goal of this course is to introduce students to security engineering. The course will help students to think as an adversary so that they can analyse systems and establish security policies. We will cover a number of common security mechanisms at all layers, and learn their properties and limitations.

Core topics:

Security design principles
Access control
Authentication mechanisms
Applied cryptography
Sotware and Network security
Privacy
Keywords
Security Privacy

Learning Prerequisites
Recommended courses
CS-233a or CS-233b Introduction to Machine Learning (for programming)
COM-208 Computer Networks
CS-323 Introduction to operating systems

Important concepts to start the course
Basic notions TCP/IP
Basic notions programming

Learning Outcomes
By the end of the course, the student must be able to:

Analyze systems for security
Decide on security mechanisms to apply
Establish a security policy
Teaching methods
Pre-recorded lectures

Practical assignemnts interactivel resolved in class using the concepts learned in the lectures

Written exercises to reaffirm the learning of the course

Practical programming homeworks to develop attacks and defenses

Expected student activities
Attending lectures, solving exercises, reading and demonstrating understanding of provided materials.

Assessment methods
Take home exams (80%)
Practical homeworks (20%)

ELECTROMAGNETICS I: TRANSMISSION LINES AND WAVES

Summary
The electrical signal is an essential vector for the transmission of information and energy. At high frequency, it manifests itself as an electromagnetic signal whose study requires the development of specific physical and mathematical models based on wave equations.

Content
1) Localized or distributed electronic components

Limitations of Localized Circuit Theory
Rise time and propagation time
Timing and propagation time
Component size and wavelength
The different types of electronic components
2) Transmission line theory: time domain

Heuristic discussion
Basic equations and solutions
Reflections on discontinuities
Resistive Terminations
Reactive Terminations
Nonlinear terminations: Bergeron diagram
Application: Time-domain reflectometry
Common transmission line parameters
3) Transmission line theory: frequency domain

Monochromatic waves and complex phasors
Short-circuit or open-circuit lines
Lines terminated by arbitrary impedance
Power Flow on a Transmission Line
Impedance matching
Smith's Abacus
Effect of losses and absorption
Two-Port Systems: S, Z, and M Settings
Keywords

Electromagnetic Signal, Distributed Circuits, Transmission Lines, Guided Electromagnetic Waves, Reflection and Transmission, Equivalent Circuits, Radio Frequency Circuits, Impedance.

Indicative prerequisites
Algebra, Analysis I and II, General Physics

Important concepts to master
Criteria for the validity of the hypothesis of localized circuits.

Properties of the electromagnetic signal: speed, frequency, wavelength.

Nature and behaviour of electromagnetic signals and waves: one-dimensional guided propagation (transmission lines), in time and frequency domain.

Multiple reflection diagrams. Time-domain reflectometry.

Notion de phaseur complexe. Abaque de Smith, Adaptation d'impédance.

Acquis de formation
A la fin de ce cours l'étudiant doit être capable de:

Analyser un système à lignes de transmission
Calculer la réponse d'un circuit distribué
Concevoir un système adapté en impédance
Utiliser un Abaque de Smith
Méthode d'enseignement
Ex cathedra with exercises in the classroom. A course material is provided on Moodle, containing the essentials of the course, as well as empty frames to allow the student to take notes of the demonstrations and examples made with the teacher. The exercises, directly related to the course, may require the use of a computer (MatLab or Mathematica).

Expected work
Active participation in the course and exercise sessions.

Assessment Methodology
Written exam.

ELECTRONICS II (FOR IC)

Summary
Master functional blocks requiring a higher level of abstraction. Realization of high-level electronic functions using operational amplifiers.

Content
Course

Amplifier models
Amplifier bandwidth
Logical families
the operational amplifier in negative reaction
The Operational Amplifier in Positive Response
N-order active filters
The Operational Amplifier and its Imperfections
The seesaws
AN and NA conversion

Exercises
As in electronics I, the student will apply many methods seen in class to solve practical exercises that can be verified with the simulation

Keywords
Amplifier, Quadrupole Pattern, Polarization, Small Signal Scheme, Filters, Bandwidth, Static Power, Dynamic Power, Slew-rate, Chebyshev, Butterworth, Schmitt Trigger, Comparator, Integrator, Differentiator, Monostable, Bistable, Astable, Signal Generator, Noise Margin, Fan-In, Fan-Out, Power Dissipated, Offset Voltage.

Required skills
Indicative prerequisites
Electronics I

Important concepts to master
Quadrupoles
The Operational Amplifier
Characteristic of a logical family
Digital-to-analog and analog-to-digital conversion
Learning outcomes
At the end of this course, the student should be able to:

Design Design Active Filters
Design Design amplifier circuits

Compare Compare the different logical families
Analyze Analyze the bandwidth of an electronic function
Design Design high-level electronic functions from operational amplifiers
Implement Implement the notion of time
Synthesize Synthesize Logic Circuits
Design Interfaces for logic and analog circuits
Méthode d'enseignement
Use common computer tools as well as those specific to their discipline.
Self-assess your level of acquired competence and plan your next learning objectives.

Assessment Methodology
Project to be submitted before the Christmas holidays and oral exam in January.

The project is divided into three parts:

Troubleshooting a deliberately faulty circuit
Analysis of a complex circuit
Design of a high-level electronic function

Introduction to quantum information processing
COM-309 / 5 credits

Teacher: Macris Nicolas

Language: English

Summary
Information is processed in physical devices. In the quantum regime the concept of classical bit is replaced by the quantum bit. We introduce quantum principles, and then quantum communications, key distribution, quantum entropy, and spin dynamics. No prior knowledge of quantum physics is required.

Content
Introduction a la mecanique quantique des systemes discrets.
- Polarization of photons, basic experiments

- Notion of quantum state, notion of measurement

- Quantum principles, notion of quantum bits, entanglement, no-cloning

- Bloch sphere

Cryptographie, Communications et Corrélations
- Secret key generation: BB1984 and B92 protocols
- Entanglement: EPR pairs
- Bell/CSCH inequalityl. Ekert protocol for a secret key generation
- Teleportaion, dense coding, distillation.

Spin and its dynamics

- Stern-Gerlach experiment, spin 1/2

- Dynamics of spin in magnetic fields, Rabi oscillations

- Manipulations of the spin and elementary quantum gates

- Introduction to the Jaynes-Cummings Model

Density matrices and Von Neumann entropy

- mixed states and entropy

- bipartite systems and entanglement entropy

- non-signalling and teleportaion revisited

Keywords
Polarization, spin, measurement, quantum bit, entanglement, key distribution, teleportaion, dense coding, Von Neumann entropy, spin dynamics.

Learning Prerequisites
Required courses
Linear algebra, basic probability

Important concepts to start the course
Vectors, matrices, eigenvalues, eigenvectors, projectors, inner product, algebraic manipulation of complex numbers, discrete probability distribution.

Learning Outcomes
By the end of the course, the student must be able to:

Describe principles of quantum physics
Illustrate quantum bits with photon polarization and spin
Explain basic communication protocols like key distribution, dense coding, teleportation
Describe how to manipulate qubits with magnetic fields
Define quantum entropies and list basic properties
Use Qiskit and/or Pennylane and NISQ devices
Teaching methods
Ex cathedra lectures, exercise session, practical implementationns typically with IBM Q machines.

Expected student activities
Participation in class, homeworks, hands-on exercises on IBM-Q.

Assessment methods
miniprojet
Graded homeworks
Final written exam

EMBEDDED SYSTEMS FIRMWARE

Withdrawal: It is not allowed to withdraw from this subject after the registration deadline.

Summary
The student will understand the architectures of the microprogrammed embedded systems, the architectures of the microprocessors, the memory hierarchy and the various Input/Output (I/O) peripherals included, using the Nintendo DS handheld platform as a case study.

Content
Introduction to Embedded Firmware
Systems - Introduction to the Nintendo DS
Physics Platform - Introduction to Compilation Software and GUI

Microprocessor and Memory
Hierarchy - Microprocessor Architecture in the Nintendo DS
- Organization of the Memory and Bus
Hierarchy - Combined C-assembler
Programming - DMA and Memory Management I/O and Peripheral
Circuit Management - Description of Peripherals and I/O in the Nintendo DS
- Simple
Interrupt Controllers - Interface and Button Management
- Keyboard
interface and management- Interface for drawing on the LCD screen and its management
- Touch screen interface and management
- Interface and sound management Gamedevelopment with the Nintendo DS
- Game functions division

- Complex
interrupt controllers - Advanced graphics
programming - Wireless connection (Bluetooth, WiFi)

Keywords
embedded systems, microprocessors, multi-core, I/O management, system development, complex interrupt controllers

Required skills
Indicative prerequisites
Programming Project, Microprogrammed Systems

Important concepts to master
Programming, microcontroller architectures

Learning outcomes
At the end of this course, the student should be able to:

Discriminate between types of I/O devices.
Create a project of the complete architecture of an embedded system with the Nintendo DS platform.
Describe the conditions for the interaction of different types of I/O devices.
Implement software to control I/O devices.
Develop collaborative projects within a team of students.
Transversal skills
Use an appropriate work methodology, organize one's work.
Self-assess your level of acquired competence and plan your next learning objectives.
Set goals and design an action plan to achieve them.
Manage your priorities.
Use common computer tools as well as those specific to their discipline.
Teaching method
Ex cathedra, and individual and group practical work

Expected work
Participation in the course, solving the exercises individually and carrying out a project in groups of two people.

Assessment Methodology
Practical evaluations, midterm exam (35% of the final grade) during the semester, oral exam at the end of the semester (65% of the final grade)

STOCHASTIC MODELS IN COMMUNICATION

Summary
The objective of this course is to master the tools of stochastic processes useful for an engineer working in the fields of communication systems, data science and computer science.

Content
Probability reminders: axioms of probability, random variable and random vector.
Some useful inequalities.
Continuous-time and discrete-time stochastic processes: second-order analysis (stationarity, ergodism, spectral density, Wiener-Khintchine relations, response of an invariant linear system to random inputs, Gaussian process, ARMA process, Wiener filters). Examples of application to simple cases of optimal detection or image processing.
Poisson process and impulsive Poisson noise. Example of application to transmissions over optical fibers.
Markov chains with discrete time. Classification of states, ergodic chains: asymptotic behavior, absorbing chains: reach time, simple random walks, branching processes. Examples of applications to the analysis of an algorithm or a distributed computer system.
Continuous-time Markov chains. Classification of states, ergodic chains: asymptotic behavior. The process of birth and death in a transitory and stationary state. Examples of application to the analysis of simple queues: definition, Little's law, M/M/1 queues... M/M/s/K, M/G/1.
Keywords
Probability, Stochastic Process, Moments, Stationarity, Gaussian Process, Poisson Process, Markov Chain, Queue.

Required skills
Mandatory Prerequisites
Linear Algebra (MATH 111 or equivalent).
Analysis I, II, III (MATH 101, 106, 203 or equivalent).
First probability course (MATH 232 or equivalent).
Circuits and Systems II (EE 205 or equivalent), or Signals and Systems (MICRO310/311 or equivalent), for the basic notions of linear (deterministic) systems theory.
Indicative prerequisites
Circuits and Systems I (EE 204 or equivalent) for the basic notions of circuit theory.
Analysis IV (MATH 204 or equivalent) for the notions of complex analysis.
Important concepts to master
Notions of linear algebra, in particular matrix operations (inversion, diagonalization, eigenvalues of a matrix).

Notions of analysis (functions of one or more real variables, sequences and series, linear ordinary differential equations).

Notions of linear systems theory (convolution, Fourier, Laplace and z transforms).

Learning outcomes
At the end of this course, the student should be able to:

Model a random system.
Analyze a problem with a random component.
Evaluate the solutions of a problem with a random component.
Teaching method
Ex cathedra (on the board), 4 hours per week.
Exercise sessions, 2 hours per week.
Expected work
In-session and at-home exercises
Assessment Methodology
Intermediate exams during the semester.
Final exam in session.

QUANTUM MECHANICS FOR NON-PHYSICISTS

Summary
This course introduces quantum mechanics to students who are interested in pursuing quantum science and technology but have not gone through the standard bachelor physics curriculum. The students will develop quantum intuition by working out numerical examples based on qubits and oscillator systems.

Content
1. Review of classical physics in the context of quantum phenomena

Planetary motion and atoms, radiation and quantization, stochastic processes and interference.

2. Mathematical language of quantum mechanics

Quantum states, operators, matrices, uncertainty, and time-evolution.

3. Basic quantum systems

Particle-in-a-box, harmonic oscillator, anharmonic oscillator, tunneling.

A quick look into stationary perturbation theory.

4. Coupled quantum systems

Entanglement, density matrix, measurement, and decoherence.

A quick look into Fermi's golden rule.

5. Exploring the quantum

Cavity quantum electrodynamics, quantum control, quantum non-demolishing measurements

6. Introduction to quantum computing

(time permitting)

Keywords
Quantum physics

Quantum information

Qubit

Learning Prerequisites
Required courses
Calculus, Linear algebra, Differential equations

Recommended courses
Complex calculus, Mechanics, Electromagnetism

Important concepts to start the course
Complex numbers

Matrices and linear algebra

Familiarity with Python

Learning Outcomes
By the end of the course, the student must be able to:

Solve basic problems in quantum mechanics
Manage self-study of modern quantum science
Teaching methods
Lectures and exercises

Expected student activities
Attend lectures and exercise sessions, do the homework

Assessment methods
Oral exam

MAKING INTELLIGENT THINGS A

Withdrawal: It is not allowed to withdraw from this subject after the registration deadline.

Summary
The course aims at teaching the prototyping of intelligent physical artifacts. It aims to solve real-world challenges by a combination of microcontroller programming, electronics, and computer -aided design and manufacturing. Student teams choose their own challeng in consultation with the teachers.

Content
The course will teach students essential skills in designing and prototyping intelligent physical artifacts, including microcontroller (such as Arduino and Raspberry PI) programming, practical electronics, and computer-aided design and manufacturing, using modern prototyping methods such as 3D printing and CNC milling.

The course will leverage and refine students' skillsets in computational thinking and in building advanced software artifacts, and aims to open new horizons for them by allowing them to explore new ways of connecting the learning outcomes of other IC courses with the physical world. A substantial emphasis will be put on engineering low-level (microcontroller-based) systems software.

The course will be structured into three phases - a first consisting of tutorials and crash courses on essential skill sets such as practical electronics and 3d printing; a second in which students individually build a precisely specified small intelligent thing under close guidance by the teaching staff; and a third - the main project phase - in which teams of students propose, design, and implement their own project.

Students will have access to a workshops and digital fabrication technologies such as laser cutters, CNC milling machines, and 3D printers through EPFL's Discovery Learning Labs. We will define a suitable format allowing all student teams to exchange insights and present progress throughout the semester; at the end of the semester there will be a public event to showcase the results of the projects.

Learning Prerequisites
Recommended courses
CS-101 Advanced ICC I
CS-173 Fundamentals of digital systems
Important concepts to start the course
Basic programming skills.

Learning Outcomes
By the end of the course, the student must be able to:

Apply a design thinking methodology in a project of inventing and prototyping an intelligent thing

Design and develop simple microcontroller-based electronic circuits with sensors and actuators
Prove constructive feedback on other groups'projects
Evaluate how to best integrate computational methods and digital fabrication tools to achieve project goals
Design a suitable format and material for public presentation of project outcomes
Assess / Evaluate own project progress and device adaptations of the project plan if necessary
Teaching methods
(Video) lectures on background technology
Hands-on tutorials on digital fabrication technologies in collaboration with the DLL
Tutoring throughout the project
Regular project critiques in a weekly forum - students will be encouraged to give each other feedback in addition to teachers' feedback.
Expected student activities
Take an entrepreneurial approach to create and develop a new idea under physical constraints such as the feasibility and cost of fabrication
Coordinate a project team and engage in collaborative problem solving
Build basic microcontroller-driven electronic circuits with sensors and actuators
Deal with resource constraints prevalent in microcontroller programming
Program sensors and actuators; implement low-level timed protocols, such as pulse-width modulation
Fabricate and evaluate prototypes using 3d printing and related technologies
Discuss project progress in class
Provide constructive criticism and feedback to other groups
Present project outcome in a public forum
Assessment methods
20% individual project grade (phase 2)
60% Team project grade (phase 3)
20% Course participation / Critiques

THE SOFTWARE ENTERPRISE - FROM IDEAS TO PRODUCTS
Remark: Special schedule

Summary
This course teaches the journey taken by software engineering teams from incipient ideas to software products that solve real problems for real people.

Content
The combination of technical and product-management skills acquired in this course will enable students to build effective software products in teams, either within an existing organization or as founders of their own startups.

Requirements and specifications
Validation, testing, and debugging
DevOps (version control, project management, issue tracking, continuous integration)
Behavior-driven and test-driven development
Development processes
Cloud-platform and mobile-platform architectures
Product architecture
Security, privacy, and data protection
Scaling to millions of users
Differentiation and value proposition/opportunity assessment
MVP and product roadmap
Business model alternatives
Intellectual property and open-source software/hardware

Learning Prerequisites
Required courses
CS-173 Fundamentals of Digital Systems (BA2)

CS-214 Software construction (BA3)

CS-202 Computer systems (BA4)

Important concepts to start the course
Must be proficient in programming in C and Python and object-oriented Java/Kotlin/Scala

Learning Outcomes
By the end of the course, the student must be able to:

Design and implement mobile and/or cloud apps
Master a variety of system design patterns
Work in and manage a team of developers
Identify opportunities for using software to solve real-world problems
Plan a software product from A-to-Z
Assess / Evaluate progress against the plan, and adapt the plan as appropriate
Manage priorities & basics of product management

Optimize the use of time and resources to achieve a given goal
Take feedback (critique) and respond in an appropriate manner
Develop auto-didact skills
Teaching methods
Ex cathedra
Recitations and workshops
Extensive team-based project
Expected student activities
Work with team members to complete a substantial project
Independently research solutions, study documentation, etc. (auto-didact)
Assessment methods
Throughout the semester (continuous control)
Grade determined based on both team and individual performance in the project
Deliverables include an implemented software product v.1 and a written product plan for v.2

4th Semester
DATA-INTENSIVE SYSTEMS
Summary
This course covers the data management system design concepts using a hands-on approach.

Content
We will cover database architecture, data models, query optimization and planning, query engine and system programming, and storage systems. As our field is evolving rapidly, the list below is not exhaustive and will be adapted to reflect current technology trends.

Entity-relationship and relational model
Relational Algebra, Calculus, and SQL
Memory and storage hierarchy
Sorting and Indexing
Query operators and optimization
Basic synchronization mechanisms: locking and latching, task coordination
Transaction management and concurrency control

Important concepts to start the course
Excellent C/C++ programming skills
Undergraduate computer science course(s) on algorithms and data structures
Learning Outcomes
By the end of the course, the student must be able to:

Identify and understand key components of database systems
Choose or critique design choices for DB software
Express application information requirements and model the data of an application
Create and design a database with a practical application in mind while justifying choices
Explore how a DBMS works
Report performance and possible optimizations for applications utilizing a DBMS
Teaching methods
Lectures, exercices, and projects

Expected student activities
Attend the lectures to ask questions and interact with the professor
Attend the exercises session to solve and discuss exercises about the recently taught material
Complete a team project that covers the practical side of the course, e.g., build a set of key components in a DBMS
Study all the material provided and recommended during classes
Take a midterm and a final exam
Assessment methods
Labs (30%)
Midterm (0%, not graded)
Final exam (70%)

THEORY OF COMPUTATION

Summary
This course constitutes an introduction to theory of computation. It discusses the basic theoretical models of computing (finite automata, Turing machine), as well as, provides a solid and mathematically precise understanding of their fundamental capabilities and limitations.

Content
Basic models of computation (finite automata, Turing machine)
Elements of computability theory (undecidability, reducibility)
Introduction to time complexity theory (P, NP and theory of NP-completeness)
Keywords
theory of computation, Turing machines, P vs. NP problem, complexity theory, computability theory, finite automata, NP-completeness

Learning Outcomes
By the end of the course, the student must be able to:

Perform a rigorous study of performance of an algorithm or a protocol
Classify computational difficulty of a decision problem
Define the notion of NP-completeness
Analyze various computation models
Design a reduction between two computational problems
Characterize different complexity classes
Explain P vs. NP problem
Transversal skills
Use a work methodology appropriate to the task.
Continue to work through difficulties or initial failure to find optimal solutions.
Teaching methods
Ex cathedra with exercises

Assessment methods
Written exam and continuous control

ARTIFICIAL INTELLIGENCE
Remark: This course will be given for the last time in the spring of 2025

Summary
Introduction to Artificial Intelligence techniques, complemented by programming exercises that show algorithms and examples of their application to practical problems.

Content
The course has three segments that deal with the 3 different forms of logical inference: deduction, abduction, and induction:

1. Knowledge representation in predicate logic, inference
algorithms 2. Expert
Systems 3. Imprecise and uncertain
reasoning 4. Search
algorithms 5. Satisfaction of Constraints
6. Diagnosis and Planning
7. Supervised Learning
8. Unsupervised learning

9. Bio-inspired learning

Required skills
Indicative prerequisites
Software construction (or former Functional programming course)

Important concepts to master
Predicate logic

Basic Algorithms

Probability theory

Programming

Learning outcomes
At the end of this course, the student should be able to:

Choosing the right type of inference for an application
Choosing the most appropriate method for a certain type of inference
Assessing the feasibility of an application of Artificial Intelligence
Select, implement, and describe deductive inference algorithms based on predicate computation
Formulate knowledge using predicate logic

Describe inference methods with imprecise and uncertain information
Choose, implement and describe algorithms for finding and satisfying constraints
Select and describe methods for diagnosis
Select, implement, and describe methods for planning
Select, implement, and describe supervised learning methods based on examples
Select, implement and describe unsupervised learning methods
Teaching method
Ex cathedra, practical work on a computer

Expected work
Course participation and exercises: 4 hours/week

Reading: 2 hours/week

Self-employment: 3 hours/week

Assessment Methodology
Intermediate test 30%, final exam 70%

COMMUNICATIONS PROJECT
Withdrawal: It is not allowed to withdraw from this subject after the registration deadline.

Summary
The course teaches the development of systems that solve real-world challenges in communications, signal processing, foundation models, robotics, and AI. Students will work in teams, construct their ideas, and either program available hardware prototypes or build their hardware or software system.

Content
The course will teach students both technical and project management skills which are essential in developing, designing, and prototyping practical systems where the underlying challenges fall in on or multiple areas with a focus on communication, signal processing, and AI, in particular foundation models and robotics.

The primary goal of this course is to give students hands-on experience with solving real-world challenges by working in teams to program different hardware platforms and ultimately build their own projects. The overall structure of the course will consist of a introductory lectures at the beginning to introduce the project and research areas in wireless radar sensing, communication, computer vision, foundational models, and robotics. The students will have time to go through the background material needed for the course and get familiar with the hardware and sensor platforms. Students will then organize into groups of 3 or 4 and propose their project using one or more of the provided hardware platforms, with the aid of the course staff. Finally, students will design and build their own project.

This class has two types of lectures.

(1) In person lectures at the beginning of the semester. After which the lecture time will be used as office hours to help students with their projects.

Lecture 1: Class Introduction
Lecture 2: Introduction to Wireless Communications & Sensing
Lecture 3: Introduction to Perceptual Robotics and Reinforcement Learning
Lecture 4: Introduction to foundation models, transformers, and multimodal learning
(2)Online lectures on background material.

Wireless Communication
Radar Signal Processing
Robotics and Reinforcement Learning

The class will support 3 hardware platforms which students can work with

We have two additional robots which students can use. However, we do not provide support for these robots and we only have one available from each type.

Unitree Go 2 EDU
MyAGV
Learning Prerequisites
Important concepts to start the course
Basic programming skills.

Teaching methods
- lectures

- Tutorials on the hardware prototypes

- Continuous supervision and tutoring

- Extensive team work and team feedback

Expected student activities
Take an entrepreneurial approach to create and develop a practical system under the given hardware constraints.
Work with team members to complete a large practical project
Independently research solutions, learn new concepts and apply them in practice.
Debug software/hardware systems.
Discuss project progress in class
Provide constructive criticism and feedback to other groups
Present project outcome in a public forum

Assessment methods
35% Individual activities grade

65% Team project grade

COMPUTER LANGUAGE PROCESSING

Summary

We teach the fundamental aspects of analyzing and interpreting computer languages, including the techniques to build compilers. You will build a working compiler from an elegant functional language into machine code using a popular backend called LLVM (https://llvm.org)

Content

See https://lara.epfl.ch/w/cc

1. Overview, source languages and run-time models

2. Review of formal languages

3. Lexical analysis

4. Syntactic analysis (parsing)

5. Name analysis

6. Type checking

7. Code generation

8. Correctness of compilers

Keywords

programming language;

compiler;

interpreter;

regular expression;

context-free grammar;

type system;

code generation;

static code analysis

Learning Prerequisites

Recommended courses

Discrete Mathematics

Theory of computation

Functional Programming

Computer architecture

Learning Outcomes
By the end of the course, the student must be able to:

Design a programming language
Construct a compiler
Coordinate development with project partner
Formulate correctness conditions for compiler
Estimate time to implement a programming language feature
Produce a working programming language implementation
Decide which language features make implementation difficult
Specify programming language and compiler functionality
Transversal skills
Assess progress against the plan, and adapt the plan as appropriate.
Evaluate one's own performance in the team, receive and respond appropriately to feedback.
Respect the rules of the institution in which you are working.
Continue to work through difficulties or initial failure to find optimal solutions.
Demonstrate a capacity for creativity.
Take feedback (critique) and respond in an appropriate manner.
Make an oral presentation.
Write a scientific or technical report.
Teaching methods
Lectures, exercises, labs

Expected student activities
Follow lectures
Project work, indepdently and under supervision of assistants
Assessment methods
The grade is based on a midterm exam (30%) as well as programming, testing, documentation, and presentation of several projects done on student's own laptops during the semester (70%).

Different groups of students may be assigned different variants of projects. There may be small but unavoidable variations in the difficulty of different variants.

ELECTROMAGNETICS II: FIELD COMPUTATION

Summary
This course deals with electromagnetism in vacuum and in continuous media. From the fundamental principles of electromagnetism, methods for solving Maxwell's equations in a vacuum and in complex material media are established.

Content
1) Vector Analysis Reminders

Vector fields and scalar fields
Coordinate systems
Gradient, divergence and rotational
2) Electromagnetic field theory

Fundamentals: Maxwell's equations, Conservation of charge, Monochromatic fields, Constitutive relations, Boundary conditions
Fundamental theorems: Poynting's theorem, Electromagnetic duality, Uniqueness of the field, Lorentz reciprocity
3) Monochromatic plane waves

Dispersion Relation
Polarization
Conductors and skin effect
Coefficients de Fresnel
Transmission Line Theory
4) Free-space radiation

Exact Solution: Vector Potential and Scalar Potential, Lorentz Gauge, Green's Function, Infinitesimal Dipole
Far-field solution
Image method
Huygens' equivalence principle
Diffraction Limit
5) Dispersive media (weather permitting)

Plasmonic materials: Drude model
Relations de Kramers-Kronig
Keywords
Electromagnetism, Field theory, Distribution of electric charges and currents, Propagation of electromagnetic waves, Radiation, Far field, Continuous media

Required skills
Mandatory Prerequisites
General Physics (Electromagnetism)

Indicative prerequisites
Analysis I, II, III

Important concepts to master
Charges and current, Electromagnetic field, Electromagnetic waves (wavelength, frequency, velocity, characteristic impedance, polarization), Radiation, polarization of matter, dissipation.

Learning outcomes
At the end of this course, the student should be able to:

Theorizing the Fundamentals of Electromagnetism
Comparing the different electromagnetic properties of materials
Calculate electric and magnetic fields radiated by a charge distribution
Teaching method
Ex cathedra with exercises in the classroom.

Expected work
Active participation in the course and exercise sessions.

Assessment Methodology
Written exam.

INTERNET ANALYTICS

Summary
Internet analytics is the collection, modeling, and analysis of user data in large-scale online services, such as social networking, e-commerce, search, and advertisement. This class explores a number of the key functions of such online services that have become ubiquitous over the past decade.

Content
The class seeks a balance between foundational but relatively basic material in algorithms, statistics, graph theory and related fields, with real-world applications inspired by the current practice of internet and cloud services.

Specifically, we look at social & information networks, recommender systems, clustering and community detection, search/retrieval/topic models, dimensionality reduction, stream computing, and online ad auctions. Together, these provide a good coverage of the main uses for data mining and analytics applications in social networking, e-commerce, social media, etc.

The course is combination of theoretical materials and weekly laboratory sessions, where we explore several large-scale datasets from the real world. For this, you will work with a dedicated infrastructure based on Hadoop & Apache Spark.

Keywords
data mining; machine learning; social networking; map-reduce; hadoop; recommender systems; clustering; community detection; topic models; information retrieval; stream computing; ad auctions

Recommended courses
Basic linear algebra

Algorithms & data structures

Important concepts to start the course
Graphs; linear algebra; Markov chains; Python

Learning Outcomes
By the end of the course, the student must be able to:

Explore real-world data from online services
Develop framework and models for typical data mining problems in online services
Analyze the efficiency and effectiveness of these models
Data-mining and machine learning techniques to concrete real-world problems
Expected student activities
Lectures with associated homeworks explore the basic models and fundamental concepts. The labs are designed to explore very practical questions based on a number of large-scale

real-world datasets we have curated for the class. The labs draw on knowledge acquired in the lectures, but are hands-on and self-contained.

Assessment methods
Project 35%, final exam 65%

INTRODUCTION TO QUANTUM COMPUTATION

Summary
The course introduces the paradigm of quantum computation in an axiomatic way. We introduce the notion of quantum bit, gates, circuits and we treat the most important quantum algorithms. We also touch upon error correcting codes. This course is independent of COM-309.

Content
Introduction to quantum computation
- Classical circuit model, reversible computation.
- Quantum bits, Hilbert space of N qubits, unitary transformations, measurement postulate.
- Quantum circuit model, universal sets of gates.
- Deutsch and Josza's problem and algorithm.

Basic algorithms
- Hidden sub-group problem and Simon's algorithm
- Mathematical parenthesis: factoring integers and period of arithmetic functions. Notions on continued fraction expansions.
- Quantum Fourier transform and the period finding algorithm
- Shor's factoring algorithm.
- Grover's search algorithm.

Error correcting codes
- Models of noise and errors.
- Shor and Steane error correcting codes.
- Stabilizer codes.

- Calderbank-Shor-Steane construction.

Keywords
Quantum computation, quantum circuits, universal gates, quantum Fourier transform, Deutsch-Josza's algorithm. Simon algorithm, Shor's algorithm, Grover's algorithm, entanglement, quantum error correction.

Learning Prerequisites
Required courses
Linear algebra course, basic probability course

Important concepts to start the course
Matrices, unitary matrices, eigenvectors, eigenvalues, inner product, algebra of complex numbers

Learning Outcomes
By the end of the course, the student must be able to:

Explain the concept of quantum algorithm on the circuit model
Describe universal gates

Describe basic quantum algorithms
Compute the evolution of a state through a circuit
Apply the measurement postulate
Manipulate algebraic expressions involving the Dirac notation
Carry out implementation on public NISQ devices
Give an example of an error correcting code
Teaching methods
Ex cathedra classes. Exercices. Use of IBM Q NISQ devices

Expected student activities
Participation in class, exercise sessions, use of IBM Q NISQ devices

Assessment methods
Midterm
Mini project on IBM Q experience
Written final exam

PARALLELISM AND CONCURRENCY IN SOFTWARE

Summary

From sensors,to smart phones,to the world's largest datacenters and supercomputers, parallelism & concurrency is ubiquitous in modern computing.There are also many forms of parallel & concurrent execution in modern platforms with varying degrees of ease of programmability,performance & efficiency.

Content

The goal of this course is to provide a deep understanding of the fundamental principles and trade-offs involved in constructing efficient parallel or concurrent software.

Performance vs. efficiency
Forms of parallelism
Communication models
Memory models
Functional parallelism
Domain-specific languages
Throughput parallelism
Data parallelism
Distributed data parallelism
Forms of concurrency
Asynchronous programming
Coroutines and futures

Learning Prerequisites

Required courses
CS-200 Computer architecture
CS-214 Software construction
Recommended courses
CS-202 Computer systems

Important concepts to start the course
Programming in C/C++, Java or Scala
Basic assembly language programming
Basic use of tools to debug

Learning Outcomes

By the end of the course, the student must be able to:

Construct parallel software
Construct concurrent software
Construct efficient software
Design software for various platforms including CPUs, accelerators and clusters

Teaching methods

Lectures
Homework
Projects

Expected student activities
Standalone homeworks
Projects in teams
Assessment methods
20% homework
30% projects
20% midterm
30% final

PRINCIPLES OF DIGITAL COMMUNICATIONS

Summary
This course is on the foundations of digital communication. The focus is on the transmission problem (rather than being on source coding).

Content
Optimal receiver for vector channels
Optimal receiver for waveform (AWGN) channels
Various signaling schemes and their performance
Efficient signaling via finite-state machines
Efficient decoding via Viterbi algorithm
Communicating over bandlimited AWGN channels
Nyquist Criterion
Communicating over passband AWGN channels

Keywords
Detection, estimation, hypothesis testing, Nyquist, bandwidth, error probability, coding, decoding, baseband, passband, AM, QAM, PSK.

Learning Prerequisites
Required courses
Signal processing for communications and Modèles stochastiques pour les communications

Important concepts to start the course
Linear algebra, probability.

Learning Outcomes
By the end of the course, the student must be able to:

Estimate the error probability of a communication link
Design a "physical layer" communication link
Implement a prototype of a "physical layer" transmitter/receiver via Matlab
Teaching methods
Ex cathedra + exercises + project. Lots of reading at home and exercises in class.

Assessment methods
With continuous control

NEUROSCIENCE FOUNDATIONS FOR ENGINEERS

Summary

This overview course bridges computational expertise with neuroscience fundamentals, aimed at fostering interdisciplinary communication and collaboration for engineering-based neuroscience programs.

Content

All content will combine experimental data and findings with computational models.

Introduction to cellular and molecular biology
Introduction to neuroscience
Neural communication
Individual neurons and small neural populations
Sensory systems and perception I - Vision
Sensory systems and perception II -Audition, Somatosensation
Learning: Neuroplasticity and Molecular Mechanisms of Learning
Motor Control I - Systems
Motor Control II - Molecular
Cognitive Neuroscience I - Sytems
Cognitive Neuroscience II - Molecular
Molecular Genetics in Neuroscience
Neuropharmacology and Drug Design
Learning Prerequisites
Important concepts to start the course
Programming

Learning Outcomes

By the end of the course, the student must be able to:

Interpret experimental data in neuroscience
Analyze experimental data in neuroscience
Describe basic concepts in biology, neuroscience
Describe basic computational tools and models in neuroscience
Integrate biological and computational concepts in neuroscience
Teaching methods
Lectures
Practical tutorials and exercises
Journal Club or Poster Presentation
Excursions
Expected student activities
Attend lectures and take notes
Participate and prepare for tutorials and exercises
Work on exercises in a group
Assessment methods
70% final exam
15% computational exercise
15% presentation exercise

MAKING INTELLIGENT THINGS B

Withdrawal: It is not allowed to withdraw from this subject after the registration deadline.

Summary
The course aims at teaching the prototyping of intelligent physical artifacts. It aims to solve real-world challenges by a combination of microcontroller programming, electronics, and computer -aided design and manufacturing. Student teams choose their own challeng in consultation with the teachers.

Content
The course will teach students essential skills in designing and prototyping intelligent physical artifacts, including microcontroller (such as Arduino and Raspberry PI) programming, practical electronics, and computer-aided design and manufacturing, using modern prototyping methods such as 3D printing and CNC milling.

The course will leverage and refine students' skillsets in computational thinking and in building advanced software artifacts, and aims to open new horizons for them by allowing them to explore new ways of connecting the learning outcomes of other IC courses with the physical world. A substantial emphasis will be put on engineering low-level (microcontroller-based) systems software.

The course will be structured into three phases - a first consisting of tutorials and crash courses on essential skill sets such as practical electronics and 3d printing; a second in which students individually build a precisely specified small intelligent thing under close guidance by the teaching staff; and a third - the main project phase - in which teams of students propose, design, and implement their own project.

Students will have access to a workshops and digital fabrication technologies such as laser cutters, CNC milling machines, and 3D printers through EPFL's Discovery Learning Labs. We will define a suitable format allowing all student teams to exchange insights and present progress throughout the semester; at the end of the semester there will be a public event to showcase the results of the projects.

Learning Prerequisites
Recommended courses
CS-101 Advanced ICC I
CS-173 Fundamentals of digital systems
Learning Outcomes
By the end of the course, the student must be able to:

Apply a design thinking methodology in a project of inventing and prototyping an intelligent thing
Design and develop simple microcontroller-based electronic circuits with sensors and actuators
Prove constructive feedback on other groups' projects
Evaluate how to best integrate computational methods and digital fabrication tools to achieve project goals

Design a suitable format and material for public presentation of project outcomes
Assess / Evaluate own project progress and device adaptations of the project plan if necessary
Teaching methods
(Video) lectures on background technology
Hands-on tutorials on digital fabrication technologies in collaboration with the DLL
Tutoring throughout the project
Regular project critiques in a weekly forum - students will be encouraged to give each other feedback in addition to teachers' feedback.
Expected student activities
Take an entrepreneurial approach to create and develop a new idea under physical constraints such as the feasibility and cost of fabrication
Coordinate a project team and engage in collaborative problem solving
Build basic microcontroller-driven electronic circuits with sensors and actuators
Deal with resource constraints prevalent in microcontroller programming
Program sensors and actuators; implement low-level timed protocols, such as pulse-width modulation
Fabricate and evaluate prototypes using 3d printing and related technologies
Discuss project progress in class
Provide constructive criticism and feedback to other groups
Present project outcome in a public forum
Assessment methods
20% individual project grade (phase 2)
60% Team project grade (phase 3)
20% Course participation / Critiques