COMPUTER SCIENCE STUDIES
AT ÉCOLE NORMALE SUPERIEURE

University year 2007/2008
CONTENTS

1. Studies in computer science at ENS and the ENS Diploma ...................................... 5
   1.1. Presentation of the ENS Diploma ................................................................................. 5
   1.2. Computer science studies at École normale supérieure .............................................. 5

2. Persons in charge ......................................................................................................... 6

3. Objectives and opportunities ........................................................................................ 6
   3.1. Objectives of the computer science speciality of the ENS Diploma ................................ 6
   3.2. Opportunities for graduating students of the computer science speciality of the ENS Diploma ................................................................. 7

4. Conditions for admission and procedures .................................................................. 7
   4.1. Civil servant students and international division scholars ......................................... 7
   4.2. The students of ENS ................................................................................................... 7
   4.3. Application form for the computer science speciality of the ENS Diploma .............. 7

5. Registration .................................................................................................................. 8

6. Organization of the curriculum ................................................................................... 8
   6.1. First year ....................................................................................................................... 9
   6.2. Second year ..................................................................................................................... 9
   6.3. Third year ....................................................................................................................... 9

7. Courses for university year 2007-2008 ...................................................................... 10
   7.1. First year: licence (L3) ................................................................................................. 10
       7.1.1 First semester .......................................................................................... 10
       7.1.2 Second semester ............................................................................................ 11
       7.1.3 Computer science internship ............................................................................... 13
   7.2. Second year: Masters (M1) .......................................................................................... 13
       7.2.1 First semester .......................................................................................... 13
       7.2.2 Second semester ............................................................................................ 16
   7.3. Third year: Masters (M2) ............................................................................................ 16
       7.3.1 First semester .......................................................................................... 16
       7.3.2 Second semester: internship ............................................................................... 16

8. Computer science courses of the ENS Diploma (outside the computer science speciality) ................................................................. 16
   8.1. Multidisciplinar speciality : mathematics and computer science ............................ 16
       8.1.1 First semester : ............................................................................................ 17
       8.1.2 Second semester : ............................................................................................ 17
       8.1.3 Third semester : .............................................................................................. 17
   8.2. Computer science as a « secondary speciality » of the ENS Diploma .............. 17
### 8.3. Computer science in the ENS Diploma

#### 9. Details of the courses for the academic year 2007/2008

<table>
<thead>
<tr>
<th>Course Description</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>Abstract interpretation: application to verification and static analysis</td>
<td>18</td>
</tr>
<tr>
<td>Advanced Complexity</td>
<td>19</td>
</tr>
<tr>
<td>Algebra 1</td>
<td>20</td>
</tr>
<tr>
<td>Algorithmic computer vision</td>
<td>20</td>
</tr>
<tr>
<td>Algorithmic Number Theory</td>
<td>21</td>
</tr>
<tr>
<td>Algorithms and programming</td>
<td>22</td>
</tr>
<tr>
<td>C Programming for the Non-computer Scientist</td>
<td>23</td>
</tr>
<tr>
<td>Communication networks</td>
<td>23</td>
</tr>
<tr>
<td>Computational geometry</td>
<td>24</td>
</tr>
<tr>
<td>Computer science logic</td>
<td>25</td>
</tr>
<tr>
<td>Cryptographic protocols: formal and computational proofs</td>
<td>26</td>
</tr>
<tr>
<td>Cryptology</td>
<td>27</td>
</tr>
<tr>
<td>Databases</td>
<td>28</td>
</tr>
<tr>
<td>Digital Systems: Algorithms, Microcode and Hardware</td>
<td>29</td>
</tr>
<tr>
<td>Digital systems: from algorithms to circuits</td>
<td>31</td>
</tr>
<tr>
<td>Discrete and computational geometry</td>
<td>32</td>
</tr>
<tr>
<td>Formal languages, theory of computation, complexity and algorithm analysis</td>
<td>33</td>
</tr>
<tr>
<td>Foundations of abstract interpretation</td>
<td>34</td>
</tr>
<tr>
<td>Fundamental objects and techniques in computational geometry</td>
<td>35</td>
</tr>
<tr>
<td>Geometry and computer vision</td>
<td>36</td>
</tr>
<tr>
<td>Information Theory</td>
<td>37</td>
</tr>
<tr>
<td>Initiation to cryptology</td>
<td>37</td>
</tr>
<tr>
<td>Integration and probability</td>
<td>38</td>
</tr>
<tr>
<td>Lambda-calculi and domains</td>
<td>39</td>
</tr>
<tr>
<td>Logic</td>
<td>39</td>
</tr>
<tr>
<td>Markov chains, Poisson processes and Gibbs fields</td>
<td>40</td>
</tr>
<tr>
<td>Mathematical bases of signal theory</td>
<td>40</td>
</tr>
<tr>
<td>Mathematical methods for neuroscience</td>
<td>40</td>
</tr>
<tr>
<td>Mathematical English course</td>
<td>41</td>
</tr>
<tr>
<td>Network dynamics and algorithms</td>
<td>42</td>
</tr>
<tr>
<td>Neurons and populations of neurons: modelling and simulation</td>
<td>42</td>
</tr>
<tr>
<td>Operating systems and computer networks</td>
<td>43</td>
</tr>
<tr>
<td>Programming languages and compilation</td>
<td>43</td>
</tr>
<tr>
<td>Robot motion planning: combinatorial issues via control theory</td>
<td>44</td>
</tr>
<tr>
<td>Scientific computing</td>
<td>45</td>
</tr>
<tr>
<td>Software engineering and distributed computing</td>
<td>46</td>
</tr>
<tr>
<td>Synchronous systems</td>
<td>46</td>
</tr>
</tbody>
</table>
1. **Studies in computer science at ENS and the ENS Diploma**

1.1. **Presentation of the ENS Diploma**

Following the harmonisation of European higher education, the Ecole Normale Supérieure has created in 2005 its own degree entitled ENS Diploma. It constitutes the pedagogical and scientific framework into which the ENS offers to set the predoctoral studies, beyond the academic curriculum. Its purpose is to offer a diversity of curricula combining an education of excellence in the main discipline and a flexible and ambitious opening in other disciplines.

The Diploma is open to students from preparatory classes (preparing for entry to Grandes Ecoles) and to students from French or foreign universities willing to receive the same education as the civil servant students (“civil servant students”) and international division scholars). The students are subject to a specific selection procedure (cf. *Conditions for admission*).

The ENS Diploma is awarded after three academic years (as a general rule\(^1\)), during which each student validates:

- a high level academic curriculum sanctioned by a Masters degree in a discipline named the “main speciality of the Diploma”. As a general rule\(^1\), this curriculum comprises the third year of the Bachelor (L3) and the two years of the Masters (M1 and M2), and each one of the three years corresponds to the validation of 60 ECTS units (European Credit Transfer System);
- additional courses validated by at least 36 ECTS units for the three years. These can (i) be chosen in the main speciality (courses followed in the discipline of the Masters), (ii) constitute the “secondary speciality of the Diploma” (coordinated courses in another discipline) or (iii) take advantage of the diversity of courses proposed by the ENS to the students.

1.2. **Computer science studies at École normale supérieure**

Within the ENS Diploma, the studies in computer science differ according to whether this discipline is the main speciality of the student or not:

- **Computer science students**: students for which computer science is the main speciality are pedagogically and scientifically linked to the computer science department. The students registered in the computer science department follow the computer science speciality of the *ENS Diploma*, which is a specific high level curriculum comprising the third year of the Bachelor (L3) and the two years of the Masters (M1 and M2).

- **Students from another scientific department**: the computer science department also offers courses to the students registered for other disciplines in other departments of the ENS. They can be arranged either as a coherent set of courses in computer science and constitute the secondary speciality of the student, or as a set of independent courses that the student validates for its Diploma degree with the consent of its tutor and the professors of these courses.

\(^1\) French or foreign students may also enter in second year, i.e., at the beginning of the Masters: the ENS Diploma is then awarded after the two years of the Masters (M1 and M2). This enrolment at intermediary level comes with exceptional measures for the additional courses (cf. Presentation of the ENS Diploma)
There are bridges between the different departments of the ENS. With the consent of the relevant direction of studies, students of the computer science speciality may be reoriented during their curriculum, either to other disciplines in the ENS Diploma, or to other academic curricula outside the Diploma.

2. Persons in charge

Director of studies:
Patrick Cousot

Assistant director of studies:
Laurent Mauborgne

Director of teaching activities:
Jean Ponce

Secretary:
Sylvia Imbert
École normale supérieure
Département d’informatique
45, rue d’Ulm – 75230 Paris cedex 05
Tel: +33 (0)1 44 32 20 45
Fax: +33 (0)1 44 32 20 75
Email: diplome@di.ens.fr

Homepage: http://diplome.di.ens.fr

3. Objectives and opportunities

3.1. Objectives of the computer science speciality of the ENS Diploma

The students registered in the computer science speciality of the Diploma are administratively linked to the computer science department of the ENS. This department offers for their main speciality an original academic curriculum with a small number of students (about twenty students per year, including the civil servant students and the international division scholars), which are trained as computer scientists with a solid background in pure and applied mathematics. Closer supervision of the students allows a faster pace and a deeper reflection than in other curricula. The courses are completed by mandatory research internships.

Objectives of the studies in computer science at ENS:

− integrating, in a top-level curriculum in computer science, civil servant students, international division scholars and students selected by the computer science department to follow their studies in the ENS Diploma (cf Conditions for admission);

− training for and by research, aiming at bringing the student to a high professional level, with top-level scientific courses. Research internships abroad, which are targeted and mandatory, open international opportunities for the student in its chosen domains;

− orienting each student according to its specific background and personal choices, thanks to a flexible structure. For this purpose, each student of the computer science speciality is
personally monitored by a tutor all along its curriculum; he is also invited to follow introductory courses in other disciplines, which customize his curriculum.

3.2. Opportunities for graduating students of the computer science speciality of the ENS Diploma

Graduates of the ENS Diploma will have a research Masters. A graduate from the computer science speciality of the Diploma can then start a PhD in mathematics or computer science, which he will achieve in principle after two or three years of research work after his curriculum. He may also work immediately after obtaining his degree.

Possible opportunities for graduates, possibly after the PhD, are as follows:

- researcher in computer science in a public research institution (CNRS, CEA, INRIA, ONERA, CNES, etc.) ;
- professor/researcher in a French or foreign university ;
- researcher in computer science in the industry (France Telecom, EADS, etc.) ;
- computer science engineer in the industry in France or abroad ;
- teacher in preparatory classes or, more generally, in higher education (IUT, CNAM, etc.)

4. Conditions for admission and procedures

4.1. Civil servant students and international division scholars

After they have obtained 120 ECTS units (L2) by following the preparatory classes and passed the competitive exam for the ENS, civil servant students wishing to follow a curriculum in computer science start their studies in computer science at ENS by registering in the first academic year (L3).

The international division scholars have also access to the studies in computer science at ENS, either at the first year level (L3) or directly at the second year level (M1), depending on their previous studies in their former university.

4.2. The students of ENS

Candidates from either a preparatory class or a French or foreign university, having obtained (directly or by equivalence) 120 ECTS units (L2) or 180 ECTS units (L3) is allowed to apply for the ENS Diploma to follow studies in computer science.

For the computer science speciality as in other disciplines within the ENS Diploma, the general rule is to select students at the third year of the Bachelor (L3). Access may however be granted at the Masters level (M1), in particular for students from foreign universities.

4.3. Application form for the computer science speciality of the ENS Diploma

The application form and conditions for application are available at the following URL:
http://www.ens.fr/entrer/diplome.php
5. Registration

The students of the computer science speciality of the Diploma must register, for each one of the three years of their curriculum, both (i) to the university degree corresponding to their current academic year and (ii) to the Diploma at ENS:

**University degree:**
- 1st year: registration at L3 at University Paris 7
- 2nd year: registration at M1 (M.P.R.I.) at ENS
- 3rd year: registration at M2 (M.P.R.I.) at ENS

**ENS Diploma:**
- All years: registration at ENS

*University degree:* after they have been selected for the studies in computer science at ENS, the students register to the third year of the Bachelor (L3) at University Paris 7, which awards them the Bachelor diploma at the end of the first year of the curriculum. During the next two years, the students register to the *Master parisien de recherche en informatique* (M.P.R.I.) at ENS, which awards them the Masters diploma at the end of their curriculum. It is also possible to register to another Masters (such as the research Masters « Mathématiques appliquées — mathématiques/vision/apprentissage » (Applied mathematics – mathematics/vision/learning) of École normale supérieure de Cachan).

*ENS Diploma:* the student registers to the « Diploma de l’École normale supérieure » every year of the curriculum. He also signs a program of studies for the current year and leaves it to the direction of studies of ENS with the signatures of his tutor and of the director of studies of the computer science department. This document defines in particular the additional courses that the student wants to validate during the current year.

6. Organization of the curriculum

The studies in computer science within the ENS Diploma are organized over three years, corresponding to the academic years L3 (Licence), M1 and M2 (Master). The academic degrees are awarded provided 60 ECTS are validated each year. At the end of the three years, the student who has passed his Masters and validated additional courses for 36 ECTS units will be awarded the Diploma of École normale supérieure. Courses validated for a national university degree (Bachelor or Masters) may not be validated a second time as an additional course of the ENS Diploma.

The additional courses of the ENS Diploma are divided in three categories:

- a language course (3 ECTS). There is a structure at ENS, named ECLA (Espace des cultures et langues d’ailleurs), specialized in teaching language courses, which allows the student to validate this course in first, second or third year of the Diploma. A mathematical English course is also offered. This language course is not mandatory if the student has done an internship lasting at least 6 months in a foreign laboratory during his diploma.

- mandatory courses of the computer science speciality (cf. *Organization of the curriculum*), representing 12 ECTS units. These two courses must be chosen among the M1 and M2 courses of the Diploma which have not been already chosen for the M.P.R.I, or among M1 or M2 mathematics courses.

- freely chosen courses, including at least 12 ECTS units outside computer science. These can be chosen among the courses offered by the computer science department or by any
other department of ENS, but also in other university curricula, with the agreement of their
tutor, of the tutor and of the director of studies of the computer science department.
Students are strongly advised to validate additional courses for at least 12 ECTS every year.

6.1. First year

The licence (L3) is awarded by University Paris 7, it requires 60 ECTS, divided in 54 ECTS of
courses at levels L3 (first and second semester) and M1 (second semester), and 6 ECTS of research
internship. These courses are organized and taught at ENS. They are regularly renewed so as to
closely follow the advancement of science. The diversity of the topics and teachers allows many
different future opportunities for the students.

The students also validate additional courses for the ENS Diploma (at least 12 ECTS
recommended every year). Additional courses in computer science at level M1 are offered during the
second semester.

Research internships in laboratories (in universities or in the industry) are planned at the end of
the first year for students, priority being given to locations in France outside Paris. A the end of the
first year, the board of examiners, within the partnership with University Paris 7, decides on
awarding the Bachelor diploma to the student, and on admission to the second year of the
predoctoral studies in computer science.

6.2. Second year

The second year students can choose between:

1. A year consisting, during the first semester, of M2 courses for 30 ECTS, and during the
second semester, of a research internship lasting at least 6 months in a foreign laboratory, for 30
ECTS. In parallel, mini-courses of research level are taught by specialists (most often foreign
professors invited by ENS). The students also validate additional courses for the ENS Diploma (at
least 12 ECTS recommended every year).

2. A full year of internship abroad, within exchange agreements between ENS and prestigious
foreign universities.

The board of examiners meets again at the end of the second year and decides on awarding the
first year of the Masters degree (M1) to the student, and on its admission to the third year of the
predoctoral studies in computer science.

6.3. Third year

During this third year, the student completes his Masters by following M2 level courses during
the first semester for 30 ECTS and, during the second semester, spends at least 5 months doing a
research internship in France or abroad, for 30 ECTS. The students also validate additional courses
for the ENS Diploma (at least 12 ECTS recommended every year).

At the end of the year, the student usually chooses a PhD supervisor and subject. At this level,
the students progressively integrate a research laboratory. In order to facilitate the integration in the
research field, it is often advisable to spend the whole year or part of the year in a laboratory in
France outside Paris or in a European country.

The board of examiners decides on awarding the Masters degree to the student, which is then
proposed to the direction of studies of ENS to obtain the Diploma. If the student has also validated
additional courses for 36 ECTS units over the course of his curriculum, the ENS awards him the ENS Diploma with the main speciality corresponding to the one of his Masters and, if relevant, with a secondary speciality in another discipline (cf. Presentation of the Diploma).

7. Courses for university year 2007-2008
In the list of courses below, the names of the professors are in bold face; the contact details of the teachers are given separately in the directory of the ENS Diploma; the ECTS (European Credit Transfer System) units are indicated for each course between parentheses.

7.1. First year: licence (L3)
7.1.1 First semester
The student must validate at least 6 courses during the first semester for the licence (L3). The 4 following courses are mandatory.

Algorithms and programming
J. Stern, D. Vergnaud, P.-A. Fouque (6 ECTS)
This course deals with the fundamental concepts of algorithmics. Each 3 hours class focuses on a new problematic. Each lecture is divided into two parts, the first devoted to basic knowledge, describing the required tools, and the second to one more advanced result. These tools are put in practice during tutorials.

Programming languages and compilation
P. Cousot, L. Mauborgne (6 ECTS)
This course covers the main concepts of programming languages and their compilation, i.e. the translation of a high-level language into machine language (lexical and syntactic analysis, typing, semantic analysis, production and optimization of intermediate and machine code, static analysis, proof of correctness of a compiler).

Formal languages, theory of computation, complexity and algorithm analysis
O. Carton, P.-A. Fouque (6 ECTS)
Finite automata, rational languages, context-free grammars and languages, pushdown automata, Turing machines, theory of computation, time and space complexity.

Digital systems: from algorithms to circuits
J. Vuillemin, F. Praden (6 ECTS)
Various scientific domains are relevant to automatic information processing, from digital physics to computer science, via electronics, algebra and telecommunications. By describing several relevant examples, this course proposes a coherent approach to this diversity.

Additionally, the student must choose and validate two courses in mathematics among the 4 following courses.
Logic
P. Dehornoy, J. Levy

Besides fundamental knowledge such as ordinal arithmetic and concepts of proof and completeness, the course presents limitation results: Gödel’s incompleteness theorems, undecidability theorems in set theory.

Algebra 1
F. Loeser

Group actions; groups and geometry; noetherian rings; elementary field theory.

Integration and probability
J. Bertoin

Integration: measurable spaces, integration with respect to a measure, construction of measures, the spaces Lp, product measures, signed measures, change of variables. Probability: foundations of probability theory, independence, convergence of random variables, conditioning.

7.1.2 Second semester
The student must validate 3 courses for the licence (L3). The following course is mandatory:

Operating systems and computer networks
J. Beigbeder

System and networks programming, based on Unix: users, files and processes management; communication tools between processes, machines or via TCP/IP. Notions of security.

The student must choose and validate two M1 computer science courses, among the following courses. Each one counts 6 ECTS for the licence (L3). One of these courses may be replaced by a M1 course in mathematics.

The students are strongly encouraged to take an additional course in this list, as an additional course for the ENS Diploma.

The courses offered here are classified in Courses in common between mathematics and computer science and Computer science courses.

Courses in common between mathematics and computer science:

Algorithmic number theory
Marc Hindry


Markov chains, Poisson processes and Gibbs fields
F. Baccelli, P. Brémaud


Mathematical bases of signal theory
P. Brémaud

This course gives the Fourier theory of functions and the spectral theory of wide-sense stationary stochastic processes in a form convenient for applications in signal processing.

Scientific computing
J. Ponce, R. Brette

Classical numerical methods of scientific computing: introduction to SCILAB; polynomial interpolation; solving linear and non-linear equations; least-squares methods; solving polynomial equations; integrating ordinary differential equations; sample applications. Intended for students from scientific disciplines outside computer science familiar with elementary linear algebra, calculus and programming.

Neurons and populations of neurons: modelling and simulation
O. Faugeras, R. Brette

Neurons communicate with each other by spikes, short discharges of electrical activity. This course addresses the following questions. How are these spikes generated? How can neurons transmit information? How does the connectivity of a population of neurons influence the type of activity it produces, and conversely, how does spike generation influence the connectivity of a population?

Computer science courses:

Databases
S. Abiteboul

This course is an introduction to databases, insisting on the relational model and with openings on the Web. The subjects that are covered include: query languages, access structures, query optimization, transaction management, distributed databases. Labs will focus on databases for Web applications.

Initiation to cryptology
J. Stern, P-A Fouque, D. Naccache

This course aims at students interested in mathematical and practical aspects of algorithmics. Its goal is to teach the fundamentals of cryptology and the main tools that are used to solve security problems.

Discrete and computational geometry
M. Pocchiola
This course introduces the objects, techniques and applications of algorithmic geometry and deals mainly with the study of line arrangements, geometric range searching and polytopes.

**Computer science logic**  
**J. Goubault-Larrecq, F. Jacquemard**  
(6 ECTS)

This course explores the basics of lambda-calculus, a fundamental tool both in computer science and logic – these two domains having strong links, via the Curry-Howard isomorphism.

**Software engineering and distributed computing**  
**J. Vermorel**  
(6 ECTS)

This course presents the fundamental concepts underlying software engineering, with a particular interest for complex and/or distributed systems. The course is associated to a software development project realized by the students organized in teams. Each session includes a regular talk followed by a collective evaluation of the student project status.

**Digital Systems: Algorithms, Microcode and Hardware**  
**D. Naccache**  
(6 ECTS)

This module provides a cross section of the various scientific disciplines around which the design of embedded system revolves. We will successively address digital circuit design, microprogramming and the optimization of embedded algorithms.

### 7.1.3 Computer science internship

The student must do an internship lasting two to three months in a public or private research laboratory, either in France or in a European country, is offered to students in computer science between June and September 2007 (professor in charge: M. Pocchiola). This internship counts 6 ECTS for the licence (L3).

### 7.2. Second year: Masters (M1)

#### 7.2.1 First semester

The student must validate M2 courses for 30 ECTS for his Masters M1, to be chosen in the following list or in courses of the MPRI. It also possible, with the agreement of the tutor and of the director of studies, to choose courses in other Masters.

**Cryptographic protocols: formal and computational proofs**  
**B. Blanchet, S. Kremer**  
(3 ECTS)

Cryptographic protocols are distributed programs which aim at securing communications and transactions by the means of cryptographic primitives. The design of cryptographic protocols is difficult: numerous errors have been discovered in protocols after their publication. It is therefore particularly important to be able to obtain proofs that protocols are secure. Two models of the protocols have been considered: the formal model and the computational model. We shall present these two models, the associated proof techniques, and results that relate them. This course will be
an opportunity to adapt and use formal tools, such as process calculi, semantics, and logic to the particular case of the study of cryptographic protocols.

**Communication networks [This course will be offered from September 2008 only]**
F. Baccelli, A. Chaintreau, M. Lelarge  
(6 ECTS)

This course presents the main protocols and mechanisms for the control of wired and wireless communication networks (multiple access protocols, congestion control, power control, routing, scheduling), along with a set of mathematical tools to evaluate their performance (theory of queues, hybrid dynamical systems, point processes).

**Information theory [This course will be offered from September 2008 only]**
M. Lelarge  
(6 ECTS)

This course presents the main principles of information theory and its application to compression and data transmission.

**Foundations of abstract interpretation**
P. Cousot, L. Mauborgne  
(6 ECTS)

The course is an introduction to abstract interpretation, which is a theory of approximation of the semantics of discrete dynamical systems, with applications such as typing and model-checking.

**Abstract interpretation: application to verification and static analysis**
P. Cousot, R. Cousot, L. Mauborgne, M. Martel  
(6 ECTS)

This course deals with static analysis, which is the process of verifying statically (without execution) dynamic program properties (at runtime) by abstract interpretation, i.e., by approximation of their semantics (describing the possible behaviours at runtime).

**Lambda-calculi and domains**
P.-L. Curien  
(6 ECTS)

This course is concerned with the syntax and semantics of programming languages, starting from the lambda-calculus. This formalism, which was introduced in logic in the 1930's, has met computer science in the 60's, when it was used for the formal specification of programming languages such as ALGOL, or for the design of languages like LISP, Scheme, or CAML. The lambda-calculus offers a rich dictionary of correspondences between programming and logic: proofs and programs, types and formulas or specifications.

**Robot motion planning: combinatorial issues via control theory**
J.-P. Laumond, F. Lamiraux  
(6 ECTS)

Motion planning deals with algorithms to compute collision-free paths for a mechanical system (mobile robots, manipulators, virtual beings...) moving amidst obstacles. The approaches consist in exploring the so-called configuration space of the considered system.

**Cryptology**
G. Hanrot, D. Pointcheval

(6 ECTS)

The purpose of this course is to present to the students the concepts and tools of modern cryptology, in as they concern conventional cryptology as well as public key methods. We stress the role that is played in the design of cryptographic protocols, by methods coming from the algorithmics and complexity theory. We also relate cryptographic procedures to the functions that they assure in terms of security (integrity, authentication, privacy).

Fundamental objects and techniques in computational geometry
M. Pocchiola, F. Chazal

(3 ECTS)

The main objective of the course is to give the students the fundamental knowledge to tackle the literature in computational geometry.

Network dynamics and algorithms
F. Baccelli, J. Mairesse

(6 ECTS)

This course has two aims: to give the scientific foundations of the theory of network dynamics; to illustrate this theory through concrete examples in the domain of communications. Product-form Markovian networks, max-plus networks, network calculus (deterministic networks for the Internet), algorithmics of ad hoc and peer-to-peer networks, random graphs and large graphs met in practice.

Synchronous systems
M. Pouzet, J. Vuillemin

(3 ECTS)

The synchronous methodology has been used successfully for the design and implementation of safe-critical embedded systems. This course gives an introduction to synchronous systems from theoretical aspect through practical applications. It addresses both hardware and software aspects which are intrinsically related in the field. Foundations of synchronous systems and languages – Esterel and applications – Synchronous circuits.

Geometry and computer vision
J. Ponce

(3 ECTS)

This course is concerned with the geometric aspects of image formation and interpretation in computer vision. Elementary (Euclidian, projective, or differential) geometry is used as a framework for studying the characteristics of different camera types and the relations between points, planes, curved surfaces, and their projections. Applications to calibration, stereovision, and image-based object and scene modelling are presented.

Mathematical methods for neuroscience
O. Faugeras

(6 ECTS)

We present a number of mathematical tools that are central to modeling in neuroscience. The prerequisites to the course are a good knowledge of differential calculus and probability theory from the viewpoint of measure theory. The thrust of the lectures is to show the applicability to neuroscience of the mathematical concepts without giving up mathematical rigor.
Vision: 3D reconstruction
R. Keriven

(3 ECTS)

Vision: restoration and segmentation
R. Deriche, N. Paragios

(3 ECTS)

Symbolic computation
A. Bostan, B. Salvy

(6 ECTS)

7.2.2 Second semester
The student must do an internship abroad lasting at least 6 months to validate 30 ECTS for his M1 diploma.

7.3. Third year: Masters (M2)
7.3.1 First semester
The student must validate M2 courses from the MPRI for 30 ECTS. It is also possible to validate courses from another academic programme, with the agreement of the tutor and of the director of studies.

The student must also validate additional courses for the ENS Diploma, unless he/she already has validated the 36 ECTS required.

7.3.2 Second semester: internship
The student must do an internship in France or abroad, lasting at least 5 months. The internship counts 30 ECTS for the M2 diploma.

8. Computer science courses of the ENS Diploma (outside the computer science speciality)
8.1. Multidisciplinary speciality: mathematics and computer science

The multidisciplinary speciality Mathematics and Computer Science, jointly organized by the departments of mathematics and computer science of ENS, offers a double fundamental curriculum in both mathematics and computer science. It also allows the students to delay their choice between the mathematics and computer science curricula, after they have received advanced courses in both disciplines. The fundamental courses of this multidisciplinary speciality are planned over three semesters, and are described below.

The students register in licence (L3) of mathematics in first year. They can also register at the same time in the first year of Mater (M1), with the opportunity of changing disciplines between mathematics and computer science after 6 months, in particular for those who plan to prepare the aggregation in second year, e.g. the aggregation of mathematics with the computer science option. However the students should be aware that the amount of work required for this double curriculum is very important. During the second semester of the second year, the students either prepare the
aggregation (e.g. with the computer science option) or do a long research internship abroad. The students may switch to other specialities of the mathematics and of the computer science curricula after each semester.

The students of this multidisciplinary speciality obtain, at the end of the second year, the first year of the Master in mathematics if they also validated two M1 courses in mathematics or the first year of the Master in computer science (e.g. the MPRI). The planning of the fundamental first year courses in both mathematics and computer science are fully compatible.

The students must validate the following courses, and do a research internship, either as a Master thesis in the mathematics curriculum of the FIMFA or a short research internship in a computer science lab as the main computer science curriculum at the end of the first year.

8.1.1 First semester:
- Mathematics courses: Logic, Integration and probabilities and Algebra 1 (this course may be replaced by Topology and differential calculus with the agreement of the tutor).
- Computer science courses: Formal languages, theory of computation, complexity and algorithm analysis, and either Algorithms and programming or Programming languages and compilation.

8.1.2 Second semester:
- Mathematics courses: Complex analysis and spectral theory and Algorithmic number theory (one of these courses may be replaced by Algebra 1, Integration and probabilities or Topology and differential calculus).
- Computer science courses: two courses among the following ones: Mathematical bases of signal theory, Markov chains, Poisson processes and Gibbs fields, Discrete and computational geometry, Neurons and populations of neurons, Scientific computing, Computer science logic.

To obtain the licence of mathematics, the students must validate 6 courses, including at least 2 courses in mathematics (except Logic) and 2 courses in computer science.

8.1.3 Third semester:
- Mathematics courses: Applied analysis or Statistics.
- Computer science courses: 2 courses among the following ones: Information theory, Communication networks, Symbolic computation, Advanced complexity, and one course among the previous or the following ones: Lambda-calculi and domains, Mathematical methods for neuroscience, Cryptology, Geometry and computer vision.

8.2. Computer science as a “secondary speciality” of the ENS Diploma

A student registered in another speciality of the Diploma than the computer science speciality (and therefore linked to another department of ENS than the computer science department) can choose to validate a coherent set of courses in computer science, which can constitute the secondary speciality of his Diploma. This coherent set of courses must represent a total of 24 ECTS units.

8.3. Computer science in the ENS Diploma
The following two courses may be taken and validated as individual courses by a student from another department of ENS outside his main speciality.

**Scientific computing** (second semester)

J. Ponce, R. Brette

(6 ECTS)

Classical numerical methods of scientific computing: introduction to SCILAB; polynomial interpolation; solving linear and non-linear equations; least-squares methods; solving polynomial equations; integrating ordinary differential equations; sample applications. Intended for students from scientific disciplines outside computer science familiar with elementary linear algebra, calculus and programming.

**Neurons and populations of neurons: modelling and simulation** (second semester)

O. Faugeras, R. Brette

(6 ECTS)

Neurons communicate with each other by spikes, short discharges of electrical activity. This course addresses the following questions. How are these spikes generated? How can neurons transmit information? How does the connectivity of a population of neurons influence the type of activity it produces, and conversely, how does spike generation influence the connectivity of a population?

These courses are aimed at beginners in computer science without specific requisites. The students who already have notions in computer science may also choose instead a L3 course of the first semester.

The computer science department also offers a course in programming, open for all students:

**C Programming for the Non-computer Scientist** (second semester)

D. Vergnaud

This introductory course is open to students from all Science and Humanity majors. No prior knowledge of computer science is assumed. The course does not aim towards any specific application of programming but will adapt to the needs of the students. Students that will be required to program software during their research may be interested in taking this course.

9. **Details of the courses for the academic year 2007/2008**

**Abstract interpretation: application to verification and static analysis**

(Patrick Cousot, Radhia Cousot, Laurent Mauborgne, Mathieu Martel)

The static analysis of programs consists in verifying statically (without execution) dynamic program properties (at runtime).

The classes of properties to be verified are diversified ranging from safety (for example the absence of runtime errors), liveness (such as the guarantee of response to a signal) and security (for example, the confidentiality of information handled by a program).

The major difficulty to automatically prove these dynamic properties is to find inductive arguments to make the proof (for example, by induction on the number of program elementary steps). Various solutions can be considered: asking the end-user (deductive methods), using finitary models (model-checking) or compute the inductive argument by approximation of the program semantics (using the fixpoint approximation techniques of abstract interpretation).
The course explores this last technique, first recalling the bases, then exploring a number of infinitary abstractions so as to handle a great number of applications to the analysis of infinite state systems, whether emerging, classical or industrialized.

Contents of the course:
- Basics of abstract interpretation;
- Numerical abstract domains;
- Symbolic abstract domains;
- Combination and refinement of abstract domains;
- Design of an abstract interpreter by abstract interpretation;
- Static analysis of sequential, procedural, recursive and modular programs;
- Probabilistic abstract domains;
- Verification of the security of cryptographic protocols;
- Estimation of the precision of numerical software;
- Static analysis of asynchronous parallel programs;
- Static analysis of distributed programs;
- Static analysis of mobile code;
- Verification of complex systems integrated on chips;
- Verification by parameterized abstraction of predicates;
- Static verifications on real-time, safety-critical embedded control-command software;
- Practical and theoretical opened problems in static analysis by abstract interpretation, perspectives.

Bibliography:

The course also includes the reading of articles relevant to the research domain and their presentation in 25 mn plus 5 mn to answer questions, which is the format most commonly used in computer science international conferences.

**Advanced Complexity**

(Jean Goubault-Larrecq)

Complexity theory goes well beyond NP-completeness. The aim of this course is to have a look at several other fundamental complexity-theoretic constructions: space complexity, alternating machines, randomized machines. We shall see a few fascinating theorems: that alternating time is equivalent to deterministic space for example, or Shamir's IP=PSPACE theorem.
Outline:
- The polynomial hierarchy, alternating machines, PSPACE. QBF is PSPACE-complete.
- Alternating complexity classes, games. The theorems by Chandra-Kozen-Stockmeyer: AL=P, AP=PSPACE. Log-space reductions. HORNSAT is P-complete.
- Directed graph reachability is NL-complete. The Immerman-Szelepcsényi theorem: non-reachability in directed graphs is also NL-complete. So NL=coNL.
- Randomized complexity classes: RP, coRP, BPP, ZPP. Reducing the error. The P/poly class. The Bennett-Gill theorem: BPP is included in P/poly. The Karp-Lipton theorem: if NP is included in BPP then PH collapses at the second level. Sipser and Gács' theorem: BPP is at the second level of the polynomial hierarchy.
- Arthur vs Merlin games. The classes MA and AM. Babai's theorem: MA is included in AM, the Arthur vs Merlin hierarchy collapses. BP . NP = AM. Arthur vs Merlin games through alternation between the E and the existential quantifiers. Interactive proofs. GRAPH-NON-ISOMORPHISM is in IP [1]. AM is at the second level of the polynomial hierarchy. The theorem of Goldwasser-Sipser: IP [k] is included in AM [k+1]. The Boppana-Håstad-Zachos theorem: if coNP is included in AM then PH collapses at the second level. Consequence for GRAPH-NON-ISOMORPHISM.
- Universal hashing techniques, GRAPH-NON-ISOMORPHISM is in AM (direct proof). The error can be made zero in case x is in L, for every language L in AM. AM is at the second level of the polynomial hierarchy.
- Classes with a polynomial number of rounds: ABPP, IP. Shamir's theorem: ABPP=IP=PSPACE.
- (optional) Fagin's theorem: the NP-complete problems on graphs are those definable by existential second-order formulæ.
- Approximation problems. The approximation thresholds of NODE COVER, TSP, KNAPSACK, MAXSAT. The Arora-Safra theorem: NP=PCP(O (logn), O (1)) (without proof). Equivalence of the Arora-Safra theorem with the fact that MAX3SAT cannot be approximated.

Algebra 1
(François Loeser)
- Groups, group actions. Symmetric group. Quotients and extensions of groups. Solvable and nilpotent groups. Sylow's Theorem.
- Groups and geometry: linear group, affine group projective group, orthogonal group... Quadratic forms: orthogonality, hyperbolic planes, Witt's Theorem and consequences.
- Noetherian rings, unit factorization domains, principal ideal domains. Finitely generated modules over p.i.d. Consequences for finitely generated abelian groups and matrices over a field.

Algorithmic computer vision
(Renaud Keriven)
Computer Vision involves Computer Science, Mathematics and Biology. This course focuses on the Computer Science side of Computer Vision. It can be considered as a survey of the algorithms commonly used in Computer Vision. These algorithms will be studied through their applications in image processing, medical imaging, video surveillance, image synthesis, virtual reality, etc. Individual programming projects will allow a closer study of one of these algorithms.

The presentations will be organized by application. For each application, one of several algorithms will be studied from the course and from recent research papers.

- **Image processing: from images to images.**
  - De-noising: anisotropic diffusion, diffusion on manifolds.
  - Inpainting: Partial Differential Equations' methods.
  - Texture synthesis: multi-resolution analysis and synthesis, total variation.
  - Image registration: mutual information.

- **2D vision: from images to 2D information.**
  - Object segmentation: active contours, level sets, active shapes, Markov Random Fields methods.
  - Tacking: particle filtering, non-parametric methods.
  - Video re-sampling: optical flow.
  - Face detection: wavelet methods.
  - Classification and recognition: clustering methods.

- **3D vision 3D: from images to 3D information.**
  - Where is the observer? auto-calibration.
  - Stereo-vision: graphs cuts, space carving.
  - Image stitching: from images, from videos.
  - Image based rendering: plenoptic modeling.

Links:

**Algorithmic Number Theory**

(Marc Hindry)

- Abelian groups and finite fields: Gauss sums, quadratic reciprocity law.
- Number of solutions of equations over a finite field.
- Applications: cryptography (RSA, primality test, factorisation)
- Algebraic Number Theory: cyclotomy, algebraic integer rings
- Analytic Number Theory: Dirichlet series, prime number theorem and arithmetic progression theorem
- Diophantine equations: elliptic curves, Siegel and Mordell-Weil theorems, effectivity questions

Bibliography:
I’ll make some course notes available; the following books may be useful:

- Z. Borevich, I. Shafarevich, Théorie des nombres (traduit du russe), Gauthier-Villars.

**Algorithms and programming**

(Jacques Stern)

This course deals simultaneously with the fundamentals of data structures and the principles of algorithm design, together with some more advanced topics. Students are expected to have had a minimal exposure to algorithms. Each lecture is divided into two parts, the first devoted to basic knowledge and the second to one more advanced result (or more exceptionally).

- Algorithms: design and evaluation.
  - basic course: termination, complexity, programming strategies,
  - advanced course: bin packing, dynamic memory allocation.

First part: data structures

- Sorting and hashing
  - basic course: examples of sorting algorithms, hashing, collisions, open hashing,
  - advanced course: Shell sort.
- Pattern matching
  - basic course: Rabin-Karp, Knuth-Morris-Pratt,
  - advanced course: algorithms for biocomputing.
- Trees
  - basic course: search trees, examples,
  - advanced course: binomial heaps, Fibonacci heaps.
- Graphs
  - basic course: transitive closure, connected components, shortest paths.
  - advanced course: eigenvalues and expansion graph.
- Flows
  - basic course: Ford-Fulkerson, Edmonds-Karp,
  - advanced course: unit flows, Dinic, matching algorithms.

Second part: numerical and symbolic algorithms

- Integers
  - basic course: multiplication, exponentiation,
  - advanced course: primality tests.
− Fast Fourier Transform
  o basic course: FFT, complexity.
  o advanced course: fast multiplication.
− Linear programming
  o basic course: simplex, complexity.
  o advanced course: the ellipsoid algorithm.
− Linear algebra and geometry of numbers
  o basic course: LUP decomposition, least squares.
  o advanced course: lattices, the LLL algorithm.
− Polynomial factorisation
  o basic course: polynomials with integer coefficients, gcd, binary polynomials.
  o advanced course: the algorithms of Berlekamp and Cantor-Zassenhaus.
− Systems of polynomial equations
  o basic course: basic standard algorithms.
  o advanced course: exp-space complexity.

C Programming for the Non-computer Scientist
(Damien Vergnaud)

This introductory course is open to students from all Science and Humanity majors. No prior knowledge of computer science is assumed. The course does not aim towards any specific application of programming but will adapt to the needs of the students. Students that will be required to program software during their research may be interested in taking this course.

Web page: http://www.di.ens.fr/~mine/enseignement/prog2006/

Communication networks
(François Baccelli, Augustin Chaintreau, Marc Lelarge)

[This course will be offered from September 2008 only]

This course is an introduction to communication networks which will cover the following domains:
− MAC Protocols in Local Area Networks
  o Wired networks: Aloha, Ethernet, tree protocols, TDMA, CSMA;
  o Wireless Networks: 802.11, spatial Aloha;
  o Stability and instability; throughput optimization.
− Congestion Control Protocols in Packet Networks
  o TCP and its variants (Reno, Tahoe, Vegas);
  o Representation by optimization problems; max-min fair, proportional fair allocations;
  o Analysis of AIMD protocols.
− Routing in IP Networks
  o IP Routing and Dijkstra's algorithm;
  o Scheduling in routers;
  o Multicast overlays.
− Queueing Networks and Teletraffic
  o Circuit switched networks (RTC, ATM);
Computer science studies at ENS 2007/2008

- IP Packet switched networks and IP traffic;
- Jackson, Kelly and Whittle networks;
- Erlang and Engset formulas;
- PS analysis of elastic traffic.

- Information Theory and Radio Access
  - Complements on networking aspects of information theory (multiaccess / broadcast channel);
  - Radio propagation, fading;
  - TDMA, CDMA and FDMA modes;
  - Maximal cell capacity on the uplink / downlink;
  - Power control in CDMA on the uplink / downlink; feasibility conditions.

Simulation projects will be proposed in complement to the course.

**Computational geometry**

(J.-D. Boissonnat, M. Pocchiola)

The goal of the module is to introduce the necessary basic knowledge for the reading of the current literature in computational geometry (topology, high dimensions, data flow, approximation, sampling techniques) and to introduce recent techniques in geometric modelling.

Detailed outline:

1. **Computational topology (M. Pocchiola).**
   - Homotopy on combinatorial surfaces
   - Homology, Morse theory
   - Persistence, topological simplification.

2. **High dimensional computational geometry (M. Pocchiola)**
   - The random projection method
   - Nearest neighbors, indexing and clustering
   - Data flow.

3. **Triangulations and meshes (M. Yvinec)**
   - Introduction
   - Triangulations et Voronoi diagrams: simplicial complexes, complexity of triangulations in 2D and 3D. Delaunay triangulations and Voronoi diagrams. The space of spheres. Power diagrams and regular triangulations.
   - Constrained triangulations: Constrained triangulations: existence and construction of constrained triangulations, optimality Delaunay constrained triangulations
   - Meshing: Introduction to meshes. Delaunay refinement and the Ruppert's method. Other meshing algorithms. What is a good mesh for linear interpolation and finite elements.

4. **Surface meshes (J-D. Boissonnat)**
   - Introduction
Computer science logic

(Jean Goubault-Larrecq)

This course explores the basics of the lambda-calculus, a tool invented by the logician Alonzo Church in the 1930s, and which is instrumental today both in semantics of programming languages (computer science) and in proof theory (logic).

The lectures are in common with ENS Cachan. Tradition is that it takes place at the rue d'Ulm.

The course is in three parts:

− Lambda-calculus and functional languages:
  o Lambda-calculus, operational semantics (reduction).
  o Expressiveness. Fixpoint and recursion combinators.
  o Termination problems, finite developments, confluence and parallel reductions.
Reduction strategies: by name, by need. Standardisation. Relationship with so-called lazy languages (Miranda, Haskell); by value, relationship with strict languages (Lisp, ML, others);

Models of the lambda-calculus, denotational semantics (in direct style) and relationship between semantics and machines;

Logical aspects:
- simply-typed lambda-calculus;
- Curry-Howard correspondence between the latter and proofs in propositional minimal logic;
- extension to classical logic, capture of continuations and exception handling;
- second-order typed lambda-calculus: Girard-Reynolds System F, correspondence with second-order intuitionistic logic;
- strong normalization, cut elimination.

3. Computer science aspects:
- calculi of explicit substitutions;
- graph reduction machines;
- environment machines, Krivine machines, closures.

Lecture notes are available at [http://www.lsv.ens-cachan.fr/~goubault/cours.html](http://www.lsv.ens-cachan.fr/~goubault/cours.html).

Bibliography:

**Cryptographic protocols: formal and computational proofs**

(Bruno Blanchet, Steve Kermer)

Cryptographic protocols are distributed programs which aim at securing communications and transactions by the means of cryptographic primitives. The design of cryptographic protocols is difficult: numerous errors have been discovered in protocols after their publication. It is therefore particularly important to be able to obtain proofs that protocols are secure.

Two models of the protocols have been considered: the formal model and the computational model. We shall present these two models, the associated proof techniques, and results that relate them.

This course will be an opportunity to adapt and use formal tools, such as process calculi, semantics, and logic to the particular case of the study of cryptographic protocols.

- Introduction. What is a cryptographic protocol? Examples of protocols and attacks.
- The passive case. We shall first focus on the simpler case in which the attacker can listen to transmitted messages but not send its own messages (passive attacker).
  - Formal model, based on terms; notion of formal indistinguishability on terms.
  - Computational model, indistinguishability.
  - Link between the two models: with adequate hypotheses, the security in the formal model implies the security in the computational model (result of Abadi and Rogaway).
− The active case. We handle here the case in which the attacker can both listen to transmitted messages and send its own messages.
  o Formal model: undecidability in the general case, result of decidability for a bounded number of sessions.
  o Formal model: automatic proofs for a non-bounded number of sessions. ProVerif tool.
  o Computational model; link between the two models (result of Cortier and Warinschi.
  o Direct automation of computational proofs; CryptoVerif tool.

Bibliography

Cryptology
(Guillaume Hanrot, David Pointcheval)

The purpose of this course is to present to the students the concepts and tools of modern cryptology, in as they concern conventional cryptology as well as public key methods. We stress the role that is played in the design of cryptographic protocols, by methods coming from the algorithmics and complexity theory. We also relate cryptographic procedures to the functions that they assure in terms of security (integrity, authentication, privacy).

- Algorithmics of integers and polynomials
This part of the course will start with a few reminders concerning the arithmetic of integers, finite fields, and of polynomials. The problem of primality ((ECPP, AKS) will then be approached, including certain tests of composition (Miller-Rabin) and tests of primality (ECPP,AKS). The factorization of integers will be strongly taken into account with the algorithms ECM, the quadratic sieve and the number field sieve NFS. We will conclude by dealing with the problem of discrete logarithms, first for a generic group then with the index calculus algorithm for the multiplicative group of finite fields.

- Elliptic and hyperelliptic curves
The aim of this part of the course is to give the definitions and properties of the curves over finite fields which are useful in order to construct a cryptosystem based on curves and for some algorithms on integers. The results (group law, Hasse -Weil bounds) will be accepted without
demonstration. The problem of counting the number of points will be evoked, without however going into detail. We shall list some attacks of the discret logarithm as regards to particular curves.

- Euclidean lattices
The algorithmics of Euclidean lattices is the principal attack technique in public key cryptography because of the deep connection between lattices and number theory. Lattices were thereby able to break knacksack-based cryptosystems, certain particular cases of RSA and DSA, etc. We will give the definition of integer lattices and also the difficult problems that accompany them (shortest vector, closest vector). The famous algorithm LLL that allows one to resolve those problems will be presented as well as some of its applications to cryptanalysis. LLL is in some sense a vectorial generalization of the famous Euclid algorithm for calculating the gcds. Finally we will take up the NTRU cryptosystem, the security of which resides in the difficulty of lattice problems.

- Asymmetric cryptography and Provable security
We will present the main concepts from asymmetric cryptography, but essentially with security notions to be acheived, such as privacy (with encryption), authentication (with signature). We will show how one can describe complete schemes and “prove” their security (with some concrete examples based on discrete logarithm and RSA.) We will then continue with more recent techniques, such as Identity-based cryptography. Eventually, we will present interactive cryptography with in particular the Zero-Knowledge protocols. They will be the building blocks of distributed cryptography.

- Symmetric cryptography
We will present the main primitives from symmetric cryptography:

  - Stream cipher. We recall the definitions for stream ciphers (Pseudo-Random Stream Generators) and their security model. We then will continue with RC4 (description and attacks), the properties of LFSRs, and the algorithm A5/1 (description and attack).

  - Block cipher. We recall the definitions for block ciphers (Pseudorandom permutations) and their security model. We then will continue with some design techniques (in particular those of AES), with the principals of differential and linear cryptanalysis, of cryptanalysis by saturation. We will then study the modes of operation of block cipher: modes for encryption and modes for compression (hashing or MAC).

Databases
(Serge Abiteboul)

This course is an introduction to databases, insisting on the relational model and with openings on the Web. The subjects that are covered include: query languages, access structures, query optimization, transaction management, distributed databases. Labs will focus on databases for Web applications.

- Introduction: Data bases and DBMS
- Query languages
  o Relational algebra
  o Relational calculus
  o Equivalence theorem
  o SQL
Computer science studies at ENS 2007/2008

Datalog
- Access structures
  - B-Trees
  - Hashing
- Distributed data management
  - Distributed DBMS
  - Warehouse
  - Data mediation
  - Peer-to-peer
  - Distributed Hash Tables
- Query optimization
  - In centralized databases
  - In distributed databases
- Concurrency control and transactions
  - Serialisability
  - Two-phase locking
  - Timestamping
- Data management on the Web
  - XML
  - XPATH and XQuery
  - Search engines

Bibliography:
- Foundations of Databases by S. Abiteboul, R. Hull and V. Vianu, Addison-Wesley available on the Web at the start of the course for the students.

**Digital Systems: Algorithms, Microcode and Hardware**

(David Naccache)

This module provides a cross section of the various scientific disciplines around which the design of embedded system revolves. We will successively address digital circuit design, microprogramming and the optimization of embedded algorithms.

Each topic will be illustrated by examples designed to be as close as possible to real-life industrial cases.

The pedagogic objectives sought are:

1. Familiarize students with machine-level **design technologies**: hardware and microprogramming.
2. While doing so present to the students the most **common embedded algorithms** (compression, arithmetic, garbage collection, error correction) that one encounters while implementing embedded systems.
The students will have to overcome the real challenges and constraints faced by embedded system designers while putting together mission-critical systems (critical paths, gate-count, low energy consumption, execution speed, RAM consumption).

- **Mathematical circuits**
  - Multiplexer, combinatorial logic and BDDs - Binary Decision Diagrams. Registers and digital synchronous circuits.
  - Lab: design a digital clock.

- **Binary algebra**
  - Average complexity circuits. Adder, subtractor, comparator.
  - Lab: design a pocket calculator.

- **Electronic circuits**
  - Various flip-flop types, counters, PROMs, sequencers, shift registers and other components.
  - Lab: enhancement of the pocket calculator with memory functions and “target value” functions.

- **Arithmetic in silicon**
  - Adders and multipliers: serial and parallel. Optimal time/surface tradeoffs.
  - Lab: enhancement of the pocket calculator with multiplication and division functions.

- **The 68HC05 microprocessor**
  - Presentation of the 68HC05 assembly programming tools. Motorola architecture. Ports, registers, memory, ALU.
  - Lab: Write a floating point library in assembler.

- **Embedded compression**
  - Shannon theory. Entropic compression. Compression using the RLE, LZW and Huffman methods.
  - Lab: RLE, LZW and Huffman on the 68HC05.

- **Embedded error correction**
  - Error control via algebraic encoding. Viterbi codes.
  - Lab: Hamming and Reed-Solomon coders / decoders on the 68HC05.

- **Embedded arithmetic: multiplication and modular reduction**
  - Multi-precision multiplication and Montgomery / Barrett reduction. Proofs of these algorithms and work factor analyses.
  - Lab: Multi-precision multiplication and Montgomery / Barrett reduction on the 68HC05.

- **Smart Cards. Standards, structure, programming, Javacard and non-volatile memory management policies (Garbage collection).**
  - Introduction to smart cards and their applications. Dissection of a CAP file.
  - Lab: Embedded garbage collector by the mark & sweep method.

- **Contactless cards, RFID.**
  - Anti-collision strategies (how a reader receiving simultaneous messages over the air manages to individuate them).
  - Lab: Exhibition of results of all Lab sessions (with posters and demos) to internal and external researchers and students.
Digital systems: from algorithms to circuits

(Jean Vuillemin)

This course gives a vertical cut through the various scientific domains which are relevant to automatic information processing: from digital physics to computer science, via electronics, binary algebra and telecommunications. Each subject is introduced through significant examples, and these applications provide a view from a height over the subject.

- Digital Synchronous Circuits
  - Multiplexer, synchronous register and assembly rules for circuits.
  - Representing Boolean functions by Binary Decision Diagram.
  - Hardware design of a Digital Watch.

- Binary Algebra
  - Boolean Algebra and integer lattice.
  - Binary arithmetics from low to high bits: the 2-adic integers $\mathbb{Z}_2$.
  - Binary Algebra over integers, rationals and sequential functions.

- Electronic Circuits
  - cMOS Transistor. Logic gates, memories et connections. Silicon design rules.
  - Isochronous clock.
  - Silicon technology and Moore's laws.
  - Schemas and layout of a serial binary adder.

- Arithmetics on Silicon
  - Adders and Multipliers. Area/Time tradeoffs.
  - Division, squares and square-roots in $\mathbb{Z}_2$ and $\mathbb{F}_2[z]$.

- The Function of a circuit
  - Sequential and Causal functions.
  - On-line circuits
  - Clock enable.
  - Circuit verification& synthesis.

- Universal Machines
  - Microprocessor principles.
  - Software design of a Digital Watch.
  - Programmable logic: FPGA - Field Programmable Gate Array and re-configurable systems.
  - Project: design and test of a microprocessor.

- Computable Real Numbers
  - Church & Turing Thesis. Limits of computing.
  - The field of computable reals. Redundant number systems.
  - On-line real arithmetics: from high to low bits.

- Digital Physics
  - Physical measurement: sampling and quantization; A/D and D/A conversion.
  - Numerical calorimeter.
  - Straight line calorimeter.
  - A circuit for the Fast Hough Transform.

- Telecommunications
  - Shannon's theory of communication.
  - Entropic compression and Huffman's algorithm.
Computer science studies at ENS 2007/2008

- Error control convolution codes.
- CODEC circuit for Hamming codes.

- Audio and video
  - Sounds and images: numerical representations.
  - Lossy compression and just noticeable distortion.
  - MPEG1 video and MP3 sound compression standards.

**Discrete and computational geometry**

(Michel Pocchiola)

The course introduces several of the main objects, techniques, and applications of discrete and computational geometry. The topics covered by the course include hyperplane arrangements, geometric range searching, the study of convex polytopes, and the solution of linear programs.

Outline:

- Cones, polyhedra and polytopes, face lattices, hyperplane arrangements, Zone theorem, Euler-Poincaré relation, cyclic polytopes and the upper bound theorem, examples of polytopes and their applications, chirotopes and oriented matroids, algorithms.
- Realisation spaces of chirotopes, realisation spaces of polytopes, universality theorems.
- Polytopal complexes, triangulations, regular triangulations, Delaunay triangulations, Voronoi diagrams, algorithms.
- Linear programs and their combinatorial models, Hirsh conjecture, examples of applications.
- Geometric hypergraphs. Vapnik-Chervonenkis dimension, epsilon-nets, randomized partitions, cuttings, simplicial partitions, geometric range searching.

The course introduces several of the main objects, techniques, and applications of *discrete and computational geometry*. The topics covered by the course include line arrangements, geometric range searching, the study of convex polytopes, and the solution of linear programs.

**Bibliography:**

In the first section on formal languages, we study languages that can be defined by finite automata or context-free grammars.

Finite automata form a basic model for discrete systems and are widely used in algorithmics. They define a basic class of languages for which we can automatically answer very efficiently most interesting questions.

Context-free grammars or pushdown automata are more expressive than finite automata. They are successfully applied to study programming languages or natural languages.

The aim of the theory of computation is to characterize which problems can be effectively solved, i.e., automatically.

In the realm of effectively solvable problems, the notion of complexity becomes essential. We will be interested by the amount of time and of space that is needed to solve a problem. These measures are intrinsic to the problem and do not depend on a specific algorithm that solves the problem.

Finally, we present methods to analyse the amount of time and memory used by specific algorithms.

1. Automata and regular languages

2. Context-free languages

3. Theory of computation

4. Complexity
   Complexity classes. Relations between complexity classes. Reduction, hardness and completeness. Examples of NL-complete, P-complete, NP-complete, PSPACE-complete problems.

5. Algorithm analysis
   Time and space complexity. Worst case complexity or average complexity. Asymptotic complexity.

Bibliography:

Foundations of abstract interpretation
(Patrick Cousot)

Reasonings on computer systems, whether purely intellectual or automatize, all require to have models of the behaviour of these computer systems at execution. The design of such mathematical models is the subject of the semantics of programming languages.

Depending upon the properties of interest of these computer systems, the model used as the basis for reasoning must be more or less concrete or refined (when one must take all details into account) or, on the contrary, more or less abstract (in order to do global reasonings ignoring irrelevant details).

Abstract interpretation is a theory of approximation which formalizes this notion of abstraction so as to design models of computer systems from existing ones by refinement or abstraction.

When the abstraction is coarse enough to be computable, one can derive typing, model-checking and static analysis algorithms from the semantics of the computer system which automatically provide information about the runtime execution of the system, without having to execute it.

Content of the course:
- Elements of order theory;
- Elements of fixpoint theory;
- Galois connections;
- Exact fixpoint abstraction;
- Application to the design of semantics by abstraction of the trace semantics;
- Constructive approximation of fixpoints;
- Type inference for the lambda-calculus by abstract interpretation;
- Design of model checking algorithms by abstract interpretation;
- Soundness and completeness of program proof methods;
- The lattice of abstract interpretations, examples.

Bibliography:
Fundamental objects and techniques in computational geometry

(Michel Pocchiola, Frédéric Chazal)

The main objective of the course is to give the students the fundamental knowledge to tackle the literature in computational geometry.

- Background and complementary material in topology. (3h, M. Pocchiola)
- Chirotopes, line arrangements, hyperplane arrangements, oriented matroids. (6h, M. Pocchiola)
- Geometric hypergraphs, Vapnik-Chervonenkis dimension, cuttings and epsilon-nets, simplicial range searching. (6h, M. Pocchiola)
- Tools and methods for the estimation and approximation of the topology and the geometry of sampled shapes: sampling and distance functions, homotopy, medial axis approximation, notions of topological persistence. (Chazal, 9h)

Bibliography:

The goal of computer vision is to allow a computer program to automatically interpret the content of a picture (photograph, X-ray image, etc.) or a set of images (multiple photographs, video clips, etc.). It has a wide range of applications in fields such as medical imaging; computer graphics and special effects for the film, TV, and video game industries; animal and human vision modeling; robotics in hostile environments and space; surveillance and security; etc. This course focuses on a fundamental aspect of computer vision - the geometric modeling of the image formation and interpretation processes. It examines the geometric characteristics of several types of cameras, as well as the geometric constraints that relate points, lines, planes, and curved surfaces to their images in photographs and videos, and presents several application examples.

The course is intended for computer scientists, to whom it will provide a rigorous framework for developing algorithms that can be applied to computer graphics, metrology, and even anthropology; to mathematicians, to whom it will provide concrete application examples of elementary but non-trivial geometric concepts; and, more generally, to students interested by geometric aspects of animal and computer vision.

Course outline:

- General introduction.
- Euclidean cameras: pinhole perspective; parallel projection; non-standard models---spherical retinas, non-central perspective, omnidirectional cameras; intrinsic and extrinsic parameters; calibration rigs and Euclidean calibration.
- Projective cameras: elements of projective geometry and line geometry; projection and inverse projection of points and lines.
- Multiple cameras: epipolar geometry; trifocal tensor; projective calibration; applications to stereovision.
- Structure from motion: calibrated cameras and Euclidean motion; uncalibrated cameras and affine or projective motion.
- Euclidean calibration without calibration charts: Chasles’ absolute conic and its cousins; applications to scene modeling from videos.
- Smooth Euclidean surfaces and their outlines: elements of descriptive differential geometry; Koenderink’s theorem; aspect graphs.
- Smooth projective surfaces and their outlines: elements of oriented differential projective geometry; visual hulls; applications to image-based modeling.
- Perspectives.

References:

**Information Theory**
(Marc Lelarge)

[This course will be offered from September 2008 only]

- Basics:
  Entropy, mutual information, typical sequences, Fano's inequality.

- Data compression:

- Capacity of digital channels:
  Shannon's theorem, Gallager's bound.

- Modulation:
  Bayesian hypothesis testing, error probability in a gaussian channel.

- Capacity of analog channels:
  Differential entropy, Shannon's theorem for the gaussian channel.

- Shared channels:
  Multi-access channel, broadcast channel

- Sampling, quantizing:
  Nyquist's theorem, distortion theory

References:

**Initiation to cryptology**
(Jacques Stern, Pierre-Alain Fouque & David Naccache)

This course aims at students interested in mathematical and practical aspects of algorithmics. Its goal is to teach the fundamentals of cryptology and the main tools that are used to solve security problems. The course is also proposed as a level 1 course for the MPRI and therefore serves as a preparation for the level 2 course in cryptology of the MPRI.
This course consists in 6 relatively independent parts, each having 4 hours of teaching and 4 hours of exercises.

- **Introduction to cryptography** - by Jacques Stern
  - Permutations, substitutions, cryptanalysis (types of attacks).
  - Integrity, confidentiality, authenticity. One Time Pad.

- **Symmetric cryptography** - by Jacques Stern
  - Stream ciphers.
  - Block ciphers.
  - Modes of operation (CBC, ECB, CTR).
  - Examples: DES, AES, RC4, A5/1.
  - Hash, MAC.

- **Algorithmic techniques** - by Pierre-Alain Fouque
  - Algorithmic of integers.
  - Algorithmic of polynomials.
  - Modular arithmetic.
  - Finite fields.

- **Asymmetric cryptography** - by Pierre-Alain Fouque
  - RSA, Diffie-Hellman, El Gamal.
  - Multiplicativity of RSA (Hastad, attacks based on multiplicativity).
  - One-way Functions, trapdoors.
  - Pseudo random generators.
  - RSA signatures, El Gamal.

- **Protocols** - by David Naccache
  - Introduction to Zero-Knowledge proofs.
  - Identification, signatures (FS, Schnorr).

- **Applications** - by David Naccache
  - PKI, IPSEC, EMV.
  - Secure channel: SSL.
  - GSM.

**Integration and probability**

(Jean Bertoin)

- **Integration**
  - Measurable spaces
  - Integration with respect to a measure
  - Construction of measures
  - The spaces $L^p$
  - Product measures
  - Signed measures
  - Change of variables

- **Probability**
  - Foundations of probability theory
  - Independence
  - Convergence of random variables
  - Conditioning
Lambda-calculi and domains

(Pierre-Louis Curien)

This course is concerned with the syntax and semantics of programming languages, starting from the lambda-calculus. This formalism, which was introduced in logic in the 1930's, has met computer science in the 60's, when it was used for the formal specification of programming languages such as ALGOL, or for the design of languages like LISP, Scheme, or CAML. The lambda-calculus offers a rich dictionary of correspondences between programming and logic: proofs and programs, types and formulas or specifications.

We shall prove the main syntactical theorems of the lambda-calculus: confluence, standardisation, termination. Then we shall present the models of the lambda-calculus: to this aim, the language of category theory will be used.

Interpreting a language in a model is akin to a compilation, and as a matter of fact models offer occasions to return to syntax: abstract machines for program execution, proofs of properties of programs. Similarly, observations on a particular model of the lambda-calculus have lead Girard to linear logic, which has connectives allowing to control hypotheses viewed as resources.

We shall also discuss extensions of the lambda-calculus with imperative features such as references and exceptions, both at a syntactical and semantical level. Here too, categories provide suitable abstractions such as monads, which allow for a modular and uniform view of different notions of effects in programming.

Finally, links with concurrency theory (pi-calculus, join calculus) will be sketched.

Recommended reading for the course:


And also:

- Categories, types and structures. A. Asperti and G. Longo. MIT Press, 1991 (sold out, but available on the web page of G. Longo (di.ens.fr)).

For the lambda-calculus :


For category theory, read the first chapters of a book such that:


Logic

(Patrick Dehornoy)

- Set Theory
  - Axiomatization of sets, representation of mathematical objects by sets, foundational problems
  - Ordinal arithmetic, transfinite induction
  - Axiom of choice, cardinals
– Logic and proofs
  o Boolean logic, notion of proof and of semantics, completeness
  o First order logic, completeness and compactness
  o Basic model theory, Lowenheim-Skolem theorems, ultraproducts, Los theorem
– Limitation results
  o Indecidability vs. non-provability
  o Peano arithmetic, indecidability of arithmetic
  o Gödel's incompleteness theorems
– The notion of model in Set Theory
  o Standard and non-standard models
  o Relative consistency, notion of forcing
  o Inaccessible cardinals, notion of large cardinal

**Markov chains, Poisson processes and Gibbs fields**

(François Baccelli, Pierre Brémaud)

– Discrete random variables, generating functions.
– Random vectors, densities, characteristic functions.
– Borel-Cantelli lemma, strong law of large numbers.
– Discrete time Markov chains.
– Poisson point processes.
– Continuous time Markov chains.
– Gibbs fields.

**Mathematical bases of signal theory**

(Pierre Brémaud)

This course gives the Fourier theory of functions and the spectral theory of wide-sense stationary stochastic processes in a form convenient for applications in signal processing.

– Fourier transform in L1 and L2, Fourier series, Z-transform.
– Poisson summation formula, Nyquist sampling theorem.
– Wide-sense stationary stochastic processes, power spectral measure (Herglotz-Bochner theorem).
– Fourier representation of the trajectories (Cramer-Khintchine theorem).

Documents: Mathematical principles of signal processing (Springer, NY, 2002), course notes.

**Mathematical methods for neuroscience**

(Olivier Faugeras)

We present a number of mathematical tools that are central to modeling in neuroscience. The prerequisites to the course are a good knowledge of differential calculus and probability theory from the viewpoint of measure theory. The thrust of the lectures is to show the applicability to neuroscience of the mathematical concepts without giving up mathematical rigor.
The concepts presented in the lectures will be illustrated by exercise sessions and programming assignments in Scilab, Matlab, or Maple.

- Electrophysiology of neurons: conductance models and their reductions, integrate and fire models.
- Bifurcations: universal unfolding and codimension, the fold, Andronov-Hopf bifurcation, fold bifurcation of limit cycles, Bogdanov-Takens bifurcation, the Bautin bifurcation, the cusp.
- Application to neuronal excitability: classes 1, 2 and 3 of neurons, integrators versus resonators, slow modulation.
- The stochastic integral: Itô integral, Itô's formula.
- Stochastic differential equations: definitions, existence and uniqueness of the solution, linear equations. Application to integrate and fire models.
- Complements: stopping times and Feynman-Kac formulae, Fokker-Planck equation. Application to the time evolution of the membrane potential and the statistics of spike trains.

References:
- Yuri A. Kuznetsov, Elements of applied bifurcation theory.

Mathematical English course
(Philip Boalch)

This mathematical English course assumes some prior knowledge of English. The first part of the course will cover mathematical vocabulary: we will cover a great quantity of mathematical words and typical technical expressions, and practice reading formulae fluently. We will translate orally (from French to English) tables of contents of books and some mathematical exercises. In the second part of the course the students will make short impromptu presentations, for example recounting for 10 minutes in English something interesting seen in the morning courses or solving an exercise and presenting the solution, in English, at the blackboard. In the third part of the course
each student will give a short prepared talk (for 20-30 minutes), and the year will end with compositions or written translations of a variety of texts, from French to English.

**Network dynamics and algorithms**

(F. Baccelli, J. Mairesse)

This course has two aims:
- To give the scientific foundations of the theory of network dynamics;
- To illustrate this theory through concrete examples in the domain of communications.

The course is structured in 4 themes:
- *Queueing networks and Markovian models* (packet switching networks, circuit switching networks, scheduling). Professor: Alain Jean-Marie
- *Dynamics of discrete event systems* (max-plus semi-rings, inf convolutions, topical functions, timed Petri nets, heap models, etc.). Professor: Jean Mairesse
- *Flow and congestion control in communication networks* (TCP, flow regulators, network calculus, scheduling etc.). Professor: François Baccelli
- *Concrete network protocols* (routing in mobile ad hoc networks, graph of the web, page rank, peer to peer networks). Professor: Laurent Viennot

Required knowledge: discrete probabilities, Markov chains, characteristic functions, discrete event simulation, graphs and graph algorithms.

Bibliography:

**Neurons and populations of neurons: modelling and simulation**

(O. Faugeras)

Neurons communicate with each other by spikes, short discharges of electrical activity. This course addresses the following questions. How are these spikes generated? How can neurons transmit information? How does the connectivity of a population of neurons influence the type of activity it produces, and conversely, how does spike generation influence the connectivity of a population?

The course requires only mathematical concepts of undergraduate level and is designed for students in computer science, physics or mathematics interested in neuroscience problems. It is completed by tutorials and simulations on computers.
Plan: the course consists of three parts and an introduction
  o Lesson 1: neurons, populations of neurons, brain and neural coding

Part I: neuron models
  o Lesson 2: biological and formal models
  o Lesson 3: analysis of models
  o Lesson 4: noise in neurons

Part II: models of populations of neurons
  o Lesson 5: an abstraction: homogeneous and infinite population
  o Lesson 6: oscillations and synchronization
  o Lesson 7: non homogeneous population

Part III: synaptic plasticity
  o Lesson 8: hebbian learning: the biological point of view
  o Lesson 9: non-supervised and relational learning.

Operating systems and computer networks
  (Jacques Beigbeder)
  - Unix programming in C (if necessary). Debuggers (dbx, xdbx), make. Libraries;
  - shell programming;
  - Programming in Perl;
  - User handling;
  - File and disk handling;
  - Processes and signals. fork/exec primitives;
  - Synchronization and communication between processes: pipes, semaphores, shared memory, etc.;
  - The Ethernet networks, IP, UDP, TCP. Internet. Communication by sockets;
  - Communications by RPC;
  - Threads.

Programming languages and compilation
  (Patrick Cousot)
This course covers the main concepts of programming languages (data structures, control structures, modularity, etc.) and their compilation, i.e. the translation of a high-level language into machine language. Mastery of this fundamental concept in computer science makes it possible to understand programming in depth and the mechanisms of translating computer languages involved in a large number of applications. The production of a compiler for a functional mini-language in tutorial classes provides an opportunity to put into practice all the knowledge of computer science acquired so far.
  Content of the course:
  - Brief review of functional programming in OCAML.
  - General principle of the operation of a compiler.
− Regular expressions and lexical analysis. Algebraic grammars and syntactic analysis (Earley’s algorithm, bottom-up LR(k) analysis).
− Typing, polymorphism. Verification and Hindley-Milner-Damas type inference (OCAML).
− Semantic values and actions. Attribute grammars.
− Designation of objects and environments of execution. Declaration and scope of identifiers. Parameter passing methods. Recursivity, structure of the execution stack and heap in PASCAL, CAML and JAVA.
− Production and optimisation of intermediate code.
− Production of machine code, register allocation, case of a machine with pipeline. Management of cache memories.
− Certified compilation.

Notes on the course: available at URL
http://www.di.ens.fr/~cousot/cours/compilation

There are many books on compilation, frequently highly technical. The course gives a more overall picture and is therefore not based on any particular book. As a complement to the notes on the course, the following could be consulted:

Robot motion planning: combinatorial issues via control theory
(J.P. Laumond et F. Lamiraux)

Motion planning deals with algorithms to compute collision-free paths for a mechanical system (mobile robots, manipulators, virtual beings...) moving amidst obstacles. The approaches consist in exploring the so-called configuration space of the considered system: a configuration is the set of parameters required to locate all the system bodies in the 3D space. Obstacles in the workspace are then transformed into obstacles in the configurations space. Thanks to this modeling, planning the motion for 3D bodies in the 3D space is equivalent to planning the motion of a point in some non necessarily connected manifold.
− Introduction: motion planning and applications in Robotics, CAD and Graphics.
− A geometrical formulation of the Piano Mover Problem
  o Configuration space searching: three methods to solve the problem of moving a polygon amidst polygonal obstacles
  o Algebraic geometry approach: the problem is decidable.
  o Algebraic topology and the problem of motions in contact.
− Main searching methods
  o Cell decomposition. The exemple of two disks in the plane.
  o Retraction. The exemple of manipulation problem.
− Nonholonomic systems
– Chained form systems and sinusoidal controls. Flat systems and geometric approaches.
– The worked-out exemple of a mobile robot pulling a trailer.
– New approaches via random searches.

Bibliography:

**Scientific computing**

(Jean Ponce)

This course is an introduction to classical numerical methods of scientific computing. It uses a general methodology inherited from elementary linear algebra to reduce the great majority of scientific computations to the numerical resolution of \( n \) linear equations in \( n \) unknowns. This methodology is illustrated by a large number of algorithms and their application to concrete problems in domains such as statistical data analysis, computer aided design (CAD), robotics, dynamic simulation, and image processing.

The course is intended for students in computer science and mathematics (in particular for preparing the computer science option of the agregation), and for students in other scientific disciplines who are familiar with elementary notions of linear algebra, calculus, and programming as taught in undergraduate science and engineering curricula. Scilab, a computer language dedicated to scientific computing, is taught at the beginning of the course, and it is used for all programming assignments.

Course outline:
– General introduction.
– Introduction to programming in Scilab.
– Polynomial interpolation and piecewise-polynomial interpolation: Vandermonde approach; Lagrange, Newton, and Hermite polynomials; Bézier curves and splines. Applications to CAD.
– Resolution of systems of linear equations: diagonal and triangular systems; LU decomposition; sparse and banded matrices; homogeneous and non-homogeneous linear least squares. Applications to statistical data analysis.
– Resolution of systems of non-linear equations: Newton’s method; non-linear least squares - Newton, Gauss-Newton, and Levenberg-Marquardt methods. Applications to image processing.
– Resolution of systems of polynomial equations: Laguerre’s method; Sturm sequences; resultants; homotopy methods. Applications to robotics.
– Integration of ordinary differential equations: Euler’s method; Runge-Kutta method; implicit methods. Applications to dynamic simulation.
Software engineering and distributed computing

( Joannès Vermorel )

This course presents the fundamental concepts underlying software engineering, with a particular interest for complex and/or distributed systems. The course is associated to a software development project realized by the students organized in teams. Each session includes a regular talk followed by a collective evaluation of the student project status.

Pre-requisites: This course is not a computer programming course. Students should be comfortable with one or several programming languages before attending the course. Without being an absolute pre-requisite, previous participation to "Algorithms and programming" and "Digital systems: from algorithms to circuits" will be appreciated.

Software engineering is the study of the software production as an economic activity, where hardware, manpower and delays are always limited resources. The advances during the last decade within the domain of software engineering have permitted large productivity gains. The course will focus on understanding how development practices associated with appropriate tools influence (positively or negatively) the productivity of the software industry.

Distributed computing systems will be integrated in the course as objects of interest, but also, through the student team project. The motivation of this choice is twofold: first, the evolution of computing hardware is leading the market toward an "everything distributed" state; second, distributed systems tend to be very challenging both to develop and to debug.

Course notes: http://www.vermorel.com/softeng.html

References:
− AntiPatterns by William J. Brown, Raphael C. Malveau, Hays W. "Skip" McCormick, Thomas J. Mowbray
− Joel on Software: And on Diverse and Occasionally Related Matters That Will Prove of Interest to Software Developers, Designers, and Managers, and to Those Who, Whether by Good Fortune or Ill Luck, Work with Them in Some Capacity by Joel Spolsky
− Design Patterns: Elements of Reusable Object-Oriented Software by Erich Gamma, Richard Helm, Ralph Johnson, John M. Vlissides

Synchronous systems

( M. Pouzet, J. Vuillemin )

The synchronous methodology has been used successfully for the design and implementation of safe-critical embedded systems. Real examples can be found in various areas such as planes, trains, nuclear plants or mobile phones.

This course gives an introduction to synchronous systems from theoretical aspect through practical applications. It addresses both hardware and software aspects which are intrinsically related in the field.

Course outline:
− Foundations:
  Stream functions: causality, continuity, sequentiality. Kahn Process Networks, streams and clocks. Automata and circuits: equivalence and minimisation. Deterministic normal forms (Mealy/Moore) and non-deterministic (Fliess).

− Synchronous languages:

− Vérification and proof

− Synchronous circuits:

Prerequisite:
It is recommended to have followed a course on compilation and semantics of programming languages; to have elementary notions of operational semantics, on systems and circuits.

References: