Faculty of Engineering, Department of

—Computing

This publication refers to the session 2009–10. The information given, including that relating to the availability of courses, is current at the time of publication; 5 October 2009; and is subject to alteration.

© Imperial College London 2009

For details of postgraduate opportunities go to www.imperial.ac.uk/pgprospectus.
Computing

Computing facilities
The Department has main teaching laboratories providing around 200 general access PC workstations, in addition to laptop docking points. Research groups within the Department also have extensive additional computing resources. Desktop systems are connected by a state-of-the-art switched gigabit Ethernet network and supported by our growing racks of cutting-edge servers. Members of the Department are provided with free file storage, email, web space, printing, and internet access. For full details of our current systems go to www.doc.ic.ac.uk/csg.

Teaching laboratories
Teaching labs provide general login access for student work, tutorials and even exams with our pioneering Lexis system. Both Linux and Windows operating systems are supported, with an extensive range of software provided on both platforms to allow general use, as well as to support teaching and research. Specialist software like the Xilinx VLSI and FPGA design system, the Matlab maths package are installed, in addition to general software like Microsoft Office, Open Office, Visual Studio, and the very latest Linux software. Most lab software is licensed for home use. All lab machines are kept up-to-date through our rolling replacement scheme.

Games and media Laboratory
This laboratory provides access to a wide range of software and hardware related to games, vision and interaction. The equipment is used to support various taught courses and group and individual project work, but can also be used by students at other times for extracurricular projects and personal recreation.

Supercomputers
The Department supports a number of high performance computer systems. Students doing individual projects may be granted access to these facilities by their supervisors. Other specialised hardware, including General Purpose Graphics Processing Units (GPGPUs), Field Programmable Gate Arrays (FPGAs), Playstations etc. are also available.

Postgraduate study
Details of postgraduate opportunities can be found in the online Postgraduate Prospectus at www.imperial.ac.uk/pgprospectus.
Undergraduate courses

- Integrated Engineering study scheme
- BSc/MSci in Mathematics and Computer Science (see page 12)

Integrated Engineering Study Scheme

BEng Computing

MEng Computing

MEng Computing (Software Engineering)

MEng Computing (Artificial Intelligence)

MEng Computing (International Programme of Study)

MEng Computing (Games, Vision and Interaction)

MEng Computing (Computation in Biology and Medicine)

- BEng Computing provides an education and develops skills in the science and engineering of computation
- MEng Computing provides a general education and professional formation in the science and engineering of computation to an advanced level

The MEng degree courses with specialisations also provide an education and professional formation in the science and engineering of computation, as well as knowledge of the following specialised fields:

- MEng Computing (Software Engineering) provides specialised engineering education in the methods, tools, techniques and processes underlying development of large and complex software.
- MEng Computing (Artificial Intelligence) provides specialisation in artificial intelligence and knowledge engineering; that is, the development of computational and engineering models of complex cognitive and social behaviours.
- MEng Computing (Games, Vision and Interaction) specialises in technologies and algorithms for supporting the development of media and arts-related applications such as computer games, visual effects and computer-generated art.
- MEng Computing (Computation in Biology and Medicine) specialises in biology-related knowledge discovery through machine learning, the modelling of biological systems and the application of computer vision and robotics in medicine.

MEng Computing (International Programme of Study) provides development of linguistic and technical skills through a programme of engineering study in the UK and abroad at a participating institution.

All courses have a common two-year study period and there is a central spine of engineering project and design work running through all years. To give maximum flexibility students may transfer between courses within the Computing Integrated Engineering Study Scheme at any time within that period.

A substantial part of the final year is devoted to an individual project allowing detailed study of a topic relevant to the student’s chosen specialisation. Students following one of the specialised MEng courses will select some of their options from a group of advanced courses set down for that specialisation.

All MEng courses include an approved period of professional formation — this will be either an industrial placement, extended project work or placement in a European industry or university. It takes place during the period between the Easter of the third year and the start of the fourth year.

The Computing Integrated Study Scheme has been developed to satisfy requirements set by various engineering institutions and are accredited by both the British Computing Society and the Institution of Engineering and Technology.

Students following the MEng Computing (International Programme of Study) spend either the first two terms of their third year abroad or their entire fourth year abroad in a recognised higher education institution (with which Imperial College has an exchange arrangement). Currently this involves institutions in the European Union and worldwide. See www.doc.ic.ac.uk/go/computing for more details.
FIRST YEAR

Compulsory core modules
Comp.112 Hardware
Comp.113 Computer systems
Comp.120 Programming
Comp.123 Object-oriented programming
Comp.130 Databases I
Comp.140 Logic
Comp.141 Reasoning about programs
Comp.142 Discrete mathematics
Comp.145 Mathematical methods
Comp.161 Laboratory
Comp.162 Laboratory workshop I
Comp.164 Professional issues

Optional modules (students must select one)
Comp.152 Foreign language I
Comp.163 Topics in AI

SECOND YEAR

Compulsory core modules
Comp.202 Software engineering—algorithms
Comp.211 Operating systems II
Comp.212 Networks and communications
Comp.220 Software engineering—design I
Comp.221 Compilers
Comp.222 Software engineering—design II
Comp.240 Models of computation
Comp.245 Statistics
Comp.261 Laboratory II
Comp.262 Laboratory workshop II

Optional modules (students must select three)
Comp.210 Computer architecture
Comp.223 Concurrency
Comp.231 Introduction to Artificial Intelligence I
Comp.233 Computational techniques
Comp.252 Foreign language II
Comp.275 C++ Lab Lecture

1 Compulsory for MEng Computing (games vision and interaction)
THIRD YEAR

BEng Computing
Students must take all compulsory modules and take eight modules in total. In addition, students must participate in a group project and undertake a major individual project.

Compulsory modules

BS0819   Organisations and management processes
Comp.302 Software engineering—methods

Optional modules
Comp.303 Software engineering—systems verification
Comp.312 Advanced databases
Comp.317 Graphics
Comp.318 Custom computing
Comp.320 Complex systems
Comp.332 Advanced computer architecture
Comp.333 Robotics
Comp.335 Distributed systems
Comp.337 Simulation and modelling
Comp.341 Introduction to bioinformatics
Comp.343 Operations research
Comp.345 Ludic computing
Comp.352 Humanities/foreign language/business\(^*\)
Comp.395 Machine learning
Comp.430 Network security
Comp.482 Type systems for programming languages

MEng Computing
Students must take all compulsory modules and take eight modules in total. In addition, students must participate in a group project.

Compulsory modules

BS0819   Organisations and management processes
Comp.302 Software engineering—methods

Optional modules
Comp.303 Software engineering—systems verification
Comp.312 Advanced databases
Comp.317 Graphics
Comp.318 Custom computing
Comp.320 Complex systems
Comp.332 Advanced computer architecture
Comp.333 Robotics
Comp.335 Distributed systems
Comp.337 Simulation and modelling
Comp.341 Introduction to bioinformatics

\(^*\) Students may choose to study up to two optional modules offered by departments in the Faculty of Engineering, the Business School or the Department of Humanities.
Comp.343  Operations research  
Comp.345  Ludic computing  
Comp.352  Humanities/foreign language/business  
Comp.430  Network security  
Comp.482  Type systems for programming languages

MEng Computing (Software Engineering)
Students must take all compulsory modules and take eight modules in total. In addition, students must participate in a group project.

Compulsory modules
BS0819  Organisations and management processes  
Comp.302  Software engineering—methods  
Comp.303  Software engineering—systems verification  
Comp.312  Advanced databases  
Comp.335  Distributed systems  

Optional modules
Comp.317  Graphics  
Comp.318  Custom computing  
Comp.332  Advanced computer architecture  
Comp.333  Robotics  
Comp.337  Simulation and modelling  
Comp.341  Introduction to bioinformatics  
Comp.343  Operations research  
Comp.345  Ludic computing  
Comp.352  Humanities/foreign language/business  
Comp.395  Machine learning  
Comp.482  Type systems for programming languages

MEng Computing (Artificial Intelligence)
Students must take all compulsory modules and take eight modules in total. In addition, students must participate in a group project.

Compulsory modules
BS0819  Organisations and management processes  
Comp.302  Software engineering—methods  
Comp.303  Software engineering—systems verification  
Comp.395  Machine learning  

Optional modules
Comp.312  Advanced databases  
Comp.317  Graphics  
Comp.318  Custom computing  
Comp.320  Complex systems  
Comp.332  Advanced computer architecture  
Comp.333  Robotics  
Comp.335  Distributed systems  
Comp.337  Simulation and modelling  
Comp.341  Introduction to bioinformatics

1 Students may choose to study up to two optional modules offered by departments in the Faculty of Engineering, the Business School or the Department of Humanities.
Comp.343  Operations research
Comp.345  Ludic computing
Comp.352  Humanities/foreign language/business
Comp.430  Network security
Comp.482  Type systems for programming languages

MEng Computing (International Programme of Study)
Students must take all compulsory modules and take eight modules in total. In addition, students must
participate in a group project.

Compulsory modules
BS0819  Organisation and management processes
Comp.302  Software engineering—methods
Comp.352  Foreign language

Optional modules
Comp.303  Software engineering—systems verification
Comp.312  Advanced databases
Comp.317  Graphics
Comp.318  Custom computing
Comp.332  Advanced computer architecture
Comp.333  Robotics
Comp.335  Distributed systems
Comp.337  Simulation and modelling
Comp.341  Introduction to bioinformatics
Comp.343  Operations research
Comp.345  Ludic computing
Comp.395  Machine learning
Comp.430  Network security
Comp.482  Type systems for programming languages

MEng Computing (Games, vision and interaction)
Students must take all compulsory modules and take eight modules in total. In addition, students must
participate in a group project.

Compulsory modules
BS0819  Organisations and management processes
Comp.302  Software engineering—methods
Comp.317  Graphics
Comp.345  Ludic computing
Students must also study either:
Comp.332  Advanced computer architecture
Comp.395  Machine learning

Optional modules
Comp.303  Software engineering—systems verification
Comp.312  Advanced databases
Comp.317  Graphics
Comp.318  Custom computing
Comp.320  Complex systems

\(^1\) Students may choose to study up to two optional modules offered by departments in the Faculty of
Engineering, the Business School or the Department of Humanities.
Comp.332 Advanced computer architecture
Comp.333 Robotics
Comp.335 Distributed systems
Comp.337 Simulation and modelling
Comp.341 Introduction to bioinformatics
Comp.343 Operations research
Comp.345 Ludic computing
Comp.352 Humanities/foreign language/business
Comp.395 Machine learning
Comp.430 Network security
Comp.482 Type systems for programming languages

MEng Computing (Computation in biology and medicine)
Students must take all compulsory modules and take eight modules in total. In addition, students must participate in a group project.

Compulsory modules
BS0819 Organisations and management processes
Comp.302 Software engineering—methods
Comp.341 Introduction to bioinformatics
Comp.395 Machine learning
Comp.312 Advanced databases

Optional modules
Comp.303 Software engineering—systems verification
Comp.312 Advanced databases
Comp.317 Graphics
Comp.318 Custom computing
Comp.320 Complex systems
Comp.332 Advanced computer architecture
Comp.333 Robotics
Comp.335 Distributed systems
Comp.337 Simulation and modelling
Comp.341 Introduction to bioinformatics
Comp.343 Operations research
Comp.345 Ludic computing
Comp.352 Humanities/foreign language/business
Comp.395 Machine learning
Comp.430 Network security
Comp.482 Type systems for programming languages

* Students may choose to study up to two optional modules offered by departments in the Faculty of Engineering, the Business School or the Department of Humanities.
FOURTH YEAR

MEng Computing

Students must take all compulsory modules and take eight modules in total. In addition, students must undertake a project outsourcing exercise, complete an industrial placement presentation and report and produce a major individual project.

Optional modules
Comp.417 Advanced graphics and visualisation
Comp.418 Computer vision
Comp.420 Cognitive robotics
Comp.422 Computational finance
Comp.424 Machine learning and neural computation
Comp.429 Parallel algorithms
Comp.430 Network security
Comp.436 Performance analysis
Comp.437 Distributed algorithms
Comp.438 Complexity
Comp.452 Humanities/foreign language/business
Comp.461 Project outsourcing exercise
Comp.464 Industrial placement—presentation and report
Comp.470 Programme analysis
Comp.471 Advanced issues in object oriented programming
Comp.474 Multi-agent systems
Comp.475 Advanced topics in software engineering
Comp.477 Computing for optimal decisions
Comp.480 Automated reasoning
Comp.481 Models of concurrent computation
Comp.482 Type systems for programming languages
Comp.484 Quantum computing
Comp.491 Knowledge representation
Comp.493 Intelligent data and probabilistic inference
Comp.499 Modal and temporal logic

MEng Computing (Software Engineering)

Students must take all compulsory modules and take eight modules in total. In addition, students must undertake a project outsourcing exercise, complete an industrial placement presentation and report and produce a major individual project.

Compulsory modules
Comp.430 Network security
Comp.475 Advanced topics in software engineering

Optional modules
Comp.417 Advanced graphics and visualisation
Comp.418 Computer vision
Comp.420 Cognitive robotics
Comp.422 Computational finance
Comp.424 Machine learning and neural computation

---

1 Students may choose to study up to two optional modules offered by departments in the Faculty of Engineering, the Business School or the Department of Humanities.

2 These modules form part of the credit-carrying assessment, though there is no scheduled teaching time.
Undergraduate syllabuses

Comp.429 Parallel algorithms
Comp.436 Performance analysis
Comp.437 Distributed algorithms
Comp.438 Complexity
Comp.452 Humanities/foreign language/business
Comp.461 Project outsourcing exercise
Comp.464 Industrial placement—presentation and report
Comp.470 Program analysis
Comp.471 Advanced issues in object-oriented programming
Comp.474 Multi-agent systems
Comp.477 Computing for optimal decisions
Comp.478 Advanced operations research
Comp.480 Automated reasoning
Comp.481 Models of concurrent computation
Comp.482 Type systems for programming languages
Comp.484 Quantum computing
Comp.491 Knowledge representation
Comp.493 Intelligent data and probabilistic inference
Comp.499 Modal and temporal logic

MEng Computing (Artificial Intelligence)
Students must take three compulsory modules and eight modules in total. In addition, students must undertake a project outsourcing exercise, complete an industrial placement presentation and report and produce a major individual project.

Compulsory modules
Students must choose three from:
Comp.420 Cognitive robotics
Comp.474 Multi-agent systems
Comp.480 Automated reasoning
Comp.491 Knowledge representation
Comp.499 Modal and temporal logic

Optional modules
Comp.417 Advanced graphics and visualisation
Comp.418 Computer vision
Comp.422 Computational finance
Comp.424 Machine Learning and Neural Computation
Comp.429 Parallel algorithms
Comp.430 Network security
Comp.436 Performance analysis
Comp.437 Distributed algorithms
Comp.438 Complexity
Comp.452 Humanities/foreign language
Comp.461 Project outsourcing exercise
Comp.464 Industrial placement—presentation and report
Comp.470 Program analysis

Students may choose to study up to two optional modules offered by departments in the Faculty of Engineering, the Business School or the Department of Humanities.

These modules form part of the credit-carrying assessment, though there is no scheduled teaching time.
Computing 11
Comp.471 Advanced issues in object oriented programming
Comp.475 Advanced topics in software engineering
Comp.477 Computing for optimal decisions
Comp.481 Models of concurrent computation
Comp.484 Quantum computing
Comp.493 Intelligent data and probabilistic inference

MEng Computing (Games, Vision and Interaction)
Students must take all compulsory modules and take eight modules in total. In addition, students must undertake a project outsourcing exercise, complete an industrial placement presentation and report and produce a major individual project.

Compulsory modules
Comp.417 Advanced graphics and visualisation
Comp.418 Computer vision
Comp.437 Distributed algorithms

Optional modules
Comp.420 Cognitive robotics
Comp.422 Computational finance
Comp.424 Machine learning and neural computation
Comp.429 Parallel algorithms
Comp.430 Network security
Comp.436 Performance analysis
Comp.438 Complexity
Comp.452 Humanities/foreign language/business
Comp.461 Project outsourcing exercise
Comp.464 Industrial placement—presentation and report
Comp.470 Programme analysis
Comp.471 Advanced issues in object oriented programming
Comp.474 Multi-agent systems
Comp.475 Advanced topics in software engineering
Comp.477 Computing for optimal decisions
Comp.480 Automated reasoning
Comp.481 Models of concurrent computation
Comp.482 Type systems for programming languages
Comp.484 Quantum computing
Comp.491 Knowledge representation
Comp.493 Intelligent data and probabilistic inference
Comp.499 Modal and temporal logic

MEng Computing (Computation in Biology and Medicine)
Students must take three compulsory modules and select eight modules in total. In addition, students must undertake a project outsourcing exercise, complete an industrial placement presentation and report and produce a major individual project.

1 Students may choose to study up to two optional modules offered by departments in the Faculty of Engineering, the Business School or the Department of Humanities.
2 These modules form part of the credit-carrying assessment, though there is no scheduled teaching time.
**Compulsory modules**

- Comp.418 Computer vision
- Comp.424 Machine learning and neural computation
- Comp.493 Intelligent data and probabilistic inference

**Optional modules**

- Comp.417 Advanced graphics and visualisation
- Comp.420 Cognitive robotics
- Comp.422 Computational finance
- Comp.429 Parallel algorithms
- Comp.430 Network security
- Comp.436 Performance analysis
- Comp.437 Distributed algorithms
- Comp.438 Complexity
- Comp.452 Humanities/foreign language/buisness
- Comp.461 Project outsourcing exercise
- Comp.464 Industrial placement—presentation and report
- Comp.470 Programme analysis
- Comp.471 Advanced issues in object oriented programming
- Comp.474 Multi-agent systems
- Comp.475 Advanced topics in software engineering
- Comp.477 Computing for optimal decisions
- Comp.480 Automated reasoning
- Comp.481 Models of concurrent computation
- Comp.482 Type systems for programming languages
- Comp.484 Quantum computing
- Comp.491 Knowledge representation
- Comp.499 Modal and temporal logic

**MEng Computing (International Programme of Study)**

The main elements of study in the fourth year will be: an individual project, a dissertation on a technology management theme, advanced language study and two or more lecture/seminar courses. A detailed plan for the year abroad will be defined for each student in collaboration with the programme of study coordinator, the receiving institution and staff from the Department of Humanities.

---

1 Students may choose to study up to two optional modules offered by departments in the Faculty of Engineering, the Business School or the Department of Humanities.

2 These modules form part of the credit-carrying assessment, though there is no scheduled teaching time.
BSc/MSci in Mathematics and Computer Science

Two joint Honours modules in Mathematics and Computer Science are offered at Imperial. The first is a three-year course leading to the degree of BSc and the second is a four-year course leading to the degree of MSci. Both modules lead to a degree of the University of London and the Associate of the Royal College of Science (ARCS), and are taught by the Departments of Computing and Mathematics.

The two programmes are designed to give students a firm foundation in both mathematics and computer science. Their flexibility caters for students with interests in both subjects, students’ studies being divided approximately equally between the two subject areas.

All the essentials of computer science are covered with a particular emphasis on developing software and reasoning formally about it. The programmes also give a firm grounding in pure mathematics, numerical analysis and statistics.

With the spread of computing procedures and mathematical ideas into many diverse areas, there is a growing need for professionals who are well versed in both computing and mathematics and who can provide a link between experts in the two disciplines.

The modules are unit based with students taking two units from each subject area in each of the first two years. In each of the third and fourth year students may select their total of four units to support their particular interests and areas of specialisation.

Each year of the course has two terms of lectures, tutorials and structured laboratory and project work. The third term is taken up with examinations and project work.

Both the three-year and four-year course consist of a common two-year core programme, followed by a wide range of optional modules offered by the two departments. The outline structure of each year is as follows:

**FIRST YEAR**
All modules are compulsory.

*Computing*
- Comp.110 Architecture
- Comp.120 Programming
- Comp.123 Object-oriented programming
- Comp.140 Logic
- Comp.141 Reasoning about programs
- Comp.162 Laboratory workshop I
- Comp.164 Professional issues
- Comp.176 Laboratory I

*Mathematics*

1. Laboratory and project work in various computer languages including: Prolog, Turing, Haskell, 80x86 Assembler.
2. See the Department of Mathematics syllabuses for details.
SECOND YEAR

Computing

Compulsory modules
 Comp.211 Operating systems II
 Comp.220 Software engineering—design I
 Comp.261 Laboratory II
 Comp.262 Laboratory workshop II

Students choose three modules from the following:
 Comp.202 Software Engineering — algorithms
 Comp.221 Compilers
 Comp.222 Software engineering—design II
 Comp.223 Concurrency
 Comp.231 Introduction to Artificial Intelligence I
 Comp.240 Models of computation

Mathematics¹

Compulsory modules
A minimum of two of the following modules are to be taken (the selection must include M1S). For greater flexibility later, it is recommended that all three modules are taken.
 M2P2 Algebra II
 M1S Probability and statistics I
 M2AA3 Orthogonality

If two compulsory modules above are selected students choose two modules from the following options, or if all three compulsory modules above are chosen, one of the options is to be taken.
 M2PM5 Metric spaces and topology
 M2AA1 Differential equations

THIRD YEAR (BSC)

Students take seven optional modules from the departmental lists including at least two technical modules from each department. Students may substitute one humanities course from a departmental approved list for one of the seven modules. In addition, students complete an individual project from within either department. Students are given guidance in planning their third and fourth year programme as an integrated whole.

Computing
 Comp.221 Compilers
 Comp.223 Concurrency
 Comp.302 Software engineering—methods
 Comp.317 Graphics
 Comp.318 Custom computing
 Comp.335 Distributed systems
 Comp.337 Simulation and modelling
 Comp.341 Introduction to bioinformatics

¹ See the Department of Mathematics syllabuses for details.
Computing

Comp.343 Operations research
Comp.352 Humanities/foreign language
Comp.395 Machine learning
Comp.436 Performance analysis
Comp.480 Automated reasoning
Comp.526 Databases
Comp.527 Computer networks and distributed systems

Mathematics

M2PM2 Algebra II
M2AA3 Orthogonality
M2PM5 Metric spaces and topology
M2AA1 Differential equations
M3N3 Optimisation
M3N4 Computational linear algebra
M3N7 Numerical solution of ordinary differential equations
M3N10 Computational partial differential equations I
M3P8 Algebra
M3P10 Group theory
M3P11 Galois theory
M3P12 Group representation theory
M3P14 Elementary number theory
M3P15 Algebraic number theory
M3P17 Discrete mathematics
M2S1 Probability and statistics II
M2S2 Statistical modelling
M3S2 Statistical modelling
M3S3 Statistical theory II
M3S4 Applied probability I
M3S7 Statistical pattern recognition
M3S8 Time series
M3S9 Stochastic simulation
M3S10 Design of experiments and surveys
M3S11 Games, risks and decisions
M3S12 Biostatistics
M3S14 Survival methods and actuarial applications
M3S15 Monte Carlo methods in financial engineering
M3T Communicating mathematics

1 Students may choose to study up to two optional modules offered by departments in the Faculty of Engineering or the Department of Humanities.

2 See the Department of Mathematics syllabuses for details.
THIRD YEAR (MSci)
Students select eight optional modules (each one a half-unit) from the departmental lists above including three technical modules from each department. Modules M2S1 Probability and Statistics II from Mathematics, and C302 Software engineering methods from Computing are compulsory. Students complete a group project in both departments.

FOURTH YEAR
Students undertake seven half-units including seven modules selected from the fourth year departmental lists and including at least two technical modules from each department. In addition, students complete an individual project in an area of their choice.

Computing
- Comp.312 Advanced databases
- Comp.317 Graphics
- Comp.332 Advanced computer architecture
- Comp.395 Machine learning
- Comp.417 Advanced graphics and visualisation
- Comp.418 Computer vision
- Comp.420 Cognitive robotics
- Comp.429 Parallel algorithms
- Comp.430 Network security
- Comp.436 Performance analysis
- Comp.437 Distributed algorithms
- Comp.438 Complexity
- Comp.452 Humanities/foreign language¹
- Comp.461 Project outsourcing exercise²
- Comp.464 Industrial placement—presentation and report²
- Comp.470 Program analysis
- Comp.471 Advanced issues in object-oriented programming
- Comp.474 Multi-agent systems
- Comp.475 Advanced topics in software engineering
- Comp.477 Computing for optimal decisions
- Comp.480 Automated reasoning
- Comp.481 Models of concurrent computation
- Comp.482 Type systems for programming languages
- Comp.484 Quantum computing
- Comp.491 Knowledge representation
- Comp.493 Intelligent data and probabilistic inference
- Comp.499 Modal and temporal logic
- Comp.527 Computer networks and distributed systems

¹ Students may choose to study up to two optional modules offered by departments in the Faculty of Engineering or the Department of Humanities.

² These modules form part of the credit-carrying assessment, though there is no scheduled teaching time.
Mathematics

Numerical/applied analysis
M4N3 Optimisation with advanced study
M4N4 Computational linear algebra with advanced study
M4N7 Numerical solution of ordinary differential equations with advanced study
M4N10 Computational partial differential equations I with advanced study
M4N11 Computational partial differential equations II
M4N12 Nonlinear functional analysis

Pure mathematics

Geometry
M4P20 Geometry I: algebraic curves with advanced study
M4P21 Geometry II: algebraic topology with advanced study
M4P31 Riemann surfaces
M4P33 Algebraic geometry
M4P43 Algebraic topology
M4P51 Riemannian geometry
M4P52 Manifolds

Algebra and discrete mathematics
M4P8 Algebra III with advanced study
M4P10 Group theory with advanced study
M4P11 Galois theory with advanced study
M4P12 Group representation theory with advanced study
M4P17 Algebraic combinatorics with advanced study
M4P22 Enumerative combinatorics with advanced study
M4P34 Groups and representations
M4P46 Lie algebras
M4P49 Model theory

Number theory
M4P14 Elementary number theory with advanced study
M4P15 Algebraic number theory with advanced study
M4P32 Number theory: elliptic curves

Statistics
M4S1 Statistical theory I with advanced study
M4S2 Statistical modelling II with advanced study
M4S3 Statistical theory II with advanced study
M4S4 Applied probability with advanced study
M4S7 Statistical pattern recognition with advanced study
M4S8 Time series with advanced study
M4S9 Stochastic simulation with advanced study
M4S10 Design of experiments and surveys with advanced study
M4S11 Games, risks and decisions with advanced study
M4S12 Biostatistics with advanced study
M4S14 Survival models and actuarial applications with advanced study
M4S15 Monte Carlo methods in financial engineering with advanced study

1 See the Department of Mathematics syllabuses for details.
Syllabuses

Comp.110 Architecture
DR N. DULAY
Autumn and spring terms
This course is the same as Comp.113 Computer systems, without the operating systems component.
Introduction: relationship to other modules, levels of abstraction, instruction set level, hardware design level, role of the computer architect.
Data representation: binary numbers, arithmetic, octal, hex, base conversion, sign and magnitude, 1’s complement and 2’s complement, BCD, overflow, characters, ASCII/Unicode. Floating point numbers: conversion, normalisation, arithmetic operations, overflow/underflow representation errors, IEEE standard: format, arithmetic, NANs, infinity and denormalised values.
Memory organisation: registers, RAM, disks; byte and word addressing; byte ordering, alignment, banks and interleaving.
CPU organisation and operation: components of a simple CPU, instructions, machine code, fetch-execute cycle, simple assembly programming.
Pentium architecture: programming model, registers, memory models, addressing modes, arrays, records, instructions, expressions, loops, procedures.
Input and output: device types and characteristics, controllers, ports, programmed I/O, interrupts, DMA, Pentium interrupt model, traps and exceptions, simple device drivers.
Process management: states and representation, creation and termination, processes and threads.
Memory management: linking and loading, fixed and dynamic partitioning, fragmentation, virtual memory, paging, segmentation.

Comp.112 Hardware
PROFESSOR D.F. GILLIES AND DR A. LOMUSCIO
Autumn term
Boolean algebra; combinatorial logic functions; principles of CMOS semiconductor devices; CMOS logic gates; tri-state gates; adders and subtractors; bistable storage device; S-R flip-flop; D-type flip-flop; latch versus edge triggering; J-K flip-flops; registers; shift registers; counters; finite state machine design; PLA; multipliers; static and dynamic RAM; ROM; CAM; magnetic storage.
Practical Laboratory work: characteristics of discrete MOSFETs; static and dynamic behaviour of basic CMOS gate and flip-flops; use of VLSI design tools to build and test simple circuits.

Comp.113 Computer systems
DR N. DULAY DR W. KNOTTENBELT AND PROFESSOR K. LEUNG
Autumn and spring terms
Introduction: relationship to other modules, levels of abstraction, instruction set level, hardware design level, role of the computer architect.
Data representation: binary numbers, arithmetic, octal, hex, base conversion, sign and magnitude, ones complement and two’s complement, BCD, overflow, characters, ASCII/Unicode. Floating point numbers: conversion, normalisation, arithmetic operations, overflow/underflow representation errors, IEEE standard: format, arithmetic, NANs, infinity and denormalised values.
Memory organisation: registers, RAM, disks; byte and word addressing; byte ordering, alignment, banks and interleaving.
CPU organisation and operation: components of a simple CPU, instructions, machine code, fetch-execute cycle, simple assembly programming.
Pentium architecture: programming model, registers, memory models, addressing modes, arrays, records, instructions, expressions, loops, procedures.
Input and output: device types and characteristics, controllers, ports, programmed I/O, interrupts, DMA, Pentium interrupt model, traps and exceptions, simple device drivers.
Operating systems introduction: objectives and functions, layers and views, user interfaces, as a resource manager, processes

Process management: states and representation, creation and termination, processes and threads.

Process scheduling: scheduling and dispatching, algorithms.

Concurrency and synchronisation: mutual exclusion, deadlocks, starvation, locks, semaphores, monitors.

Memory management: linking and loading, fixed and dynamic partitioning, fragmentation, virtual memory, paging, segmentation.

Comp.120.1 Programming I
DR A. FIELD
Autumn term
Specification and design: English specifications, Haskell specifications of Java programs and stepwise refinement. Programming in Java: control, data-structures, simple algorithm construction, testing, debugging.

Comp 120.2 Programming II
PROFESSOR C. HANKIN
Autumn term
Specification and design: English specifications, Haskell specifications of Java programs and stepwise refinement. Programming in Java: control, data-structures, simple algorithm construction, testing, debugging.

Comp 120.3 Programming III
PROFESSOR M. SHANAHAN
Summer term
Introduction to logic programming using Prolog: syntax of terms and rules, procedural interpretation, non-determinism and search, unification and answer extraction. Type checking, list processing and arithmetic. Generate-and-test algorithms, aggregation, tree-pruning. Introduction to meta-programming.

Comp.123 Object-oriented programming
DR A. RUSSO AND DR S. UCHITEL
Object-oriented program design. Key issues related to a contemporary software development process. Adaptation of the standard five-step approach to software development (problem specification, analysis, design, implementation and testing) to the object-oriented paradigm. Object diagrams as an object-oriented design and modelling technique.

Objects, primitive data and program statements Definition and use of objects. Use of predefined classes from the Java standard library. Primitive types, operators and expressions. Procedural statements and basic control structures in Java.

Classes Issues related to writing classes and methods, such as instance data, visibility, scope, method parameters and return types, constructors, object relationships, method overloading and decomposition. Various examples of related program design and implementations. Some advanced topics such as static modifiers, and interfaces for implementing polymorphism.

Interfaces and Abstract classes Definition and use of Interfaces and Abstract classes in Java. A brief introduction to inheritance and its role in software design.

Exceptions and I/O Streams Exposure (through example programs) to: exception messages, exception propagation, and checked and unchecked exceptions; I/O streams including the IOException class; Standard Java I/O; and the keyboard class.

Vectors and Iterators.

Introducing Abstract Data Types A brief introduction to data abstraction and definition of the concept of Abstract Data Types (ADT).

Lists—definition of lists and associated operations for manipulating individual elements or entire lists, such as addition and deletion of an element, search for an element, replacement of an element in a list,
Undergraduate syllabuses

computing the length of a list, sorting a list and copying a list. Array and reference-based implementation techniques. Examples of list variations, including double linked lists, circular lists.

Stacks, queues and recursion. Definition of stacks and associated operations. Array and linked-list implementation of stacks. A brief introduction to iteration and recursion. Queue and related access operations. Array and linked-list implementation of queues. Priority queues and double-ended queues. Example applications of stacks and queues.

Trees—definition of binary trees and associated operations. Array and reference-based implementation of binary trees. Various examples of tree traversal algorithms. Binary search trees and related operations such as finding, inserting, removing an element. Illustration of the Treesort algorithm.

Heaps and hash tables. Definition of heaps and basic operations. Priority queues with heaps. Illustration of the Heapsort algorithm. An introduction to hash tables, hash function, and illustration of some of the issues related with the choice of hash functions and the use of double hashing.

Graphs. Introduction of undirected and directed graphs and implementation of graphs. Illustration of basic graph algorithms such as depth-first and breadth-first search.

Comp. 130 Databases I
MR J. JACOBSON
To introduce database systems with particular reference to the relational model, including design, query languages and update transactions. To introduce entity-relationship modelling and translation to the relational model.

Introduction to databases; data modelling, database management, data dictionary, query formulation and evaluation.

Relational databases: design, functional dependencies, keys and normal forms.

Relational database languages: relational algebra. Views integrity and security.


The module will be supported by laboratory sessions using the relational database system, INGRES, and the language SQL.

Comp. 140 Logic
PROFESSOR I. HODKINSON


Comp. 141 Reasoning about programs
PROFESSOR S. DROSSOPOULOU AND DR K. BRODA

(i) Use of pre and post conditions and loop invariants for showing correctness. (ii) Standard algorithms and reasoning about their correctness. (iii) Mathematical induction and its application to reasoning about Haskell programs.

Use of mathematical induction and structural induction to show that Haskell programs meet their specification.

Reasoning with pre and post conditions and use of the method of loop invariants to show correctness of programs.

The development of some common algorithms, including binary chop, partition and quicksort, Warshall’s algorithm and variations, string searching (including Boyer Moore algorithm).

Rigorous reasoning in the specification and design of programs.

Induction: mathematical induction, structural induction.

Formal program techniques: specification by pre and post conditions, derivation and verification of programs, invariants, proofs of program properties. Conversion of recursion to iteration.
Common algorithms as examples (e.g. binary chop, Warshall's algorithm, partition and quicksort, string searching).

**Comp. 142 Discrete mathematics**  
PROFESSOR Y. GUO AND DR I. PHILLIPS  
Autumn and spring terms.  
Basic set theory: sets, relations, functions, cardinality, inverse, composition, bijection.  
Relations: equivalence relations, orderings.  
Graphs: Basic definitions, isomorphism, Eulerian paths, graph colouring, Dijkstra's shortest path algorithm.  
Algorithm analysis: time complexity, worst case and average case analysis, optimality, orders, decision trees, recurrence relations, examples from searching and sorting.

**Comp. 145 Mathematical methods**  
DR J. BRADLEY AND DR D. PATTINSON  
Autumn term  
*Vector algebra (3) for Computer graphics, computational techniques*  
Motivated by: Computer Graphics (3D vector manipulation)  
Vector notation; vector addition, multiplication, dot and cross products; vector equations and intersections of lines/planes; vector spaces and linear in/dependence  
*Matrices/linear algebra (3) for computer graphics, performance Modelling, digital libraries, Computational techniques*  
Motivated by: Computer Graphics (3D transformations) and PageRank algorithm (Eigenvector/Eigenvalue)  
Matrix notation; three dimensional transformations (rotation, scaling, skewing); matrix addition, multiplication, determinant, inverse; linear in/dependence; solution of linear equations by Gaussian elimination; Eigenvectors, Eigenvalues, characteristic equation.  
*Calculus (2) for computational techniques (Computational finance, operations research)*  
Motivated by: Computer graphics (differential algorithms) and optimisation (maxima/minima over surfaces). Review of differentiation.  
Differentiation as limit of gradient; fundamental theorem of calculus; stationary points; partial derivatives and the chain rule; recurrence relations and solution  
*Analysis (5) for numerical algorithms, scientific programming, functional programming*  
Convergence of sequences and series (arithmetic, geometric, harmonic); comparison test/absolute convergence; power series/radii of convergence; Taylor's theorem and the Hessian; finite precision arithmetic and effect on computations; introduction to fixed point problems  
*Complex numbers (1) for computer graphics, computer vision, performance modelling*  
Complex numbers and argand diagram; curra+ibCurr, Currcis \*thetaCur, curm\exp(-i\theta)Cur notations

**Comp. 152 Foreign language I (optional)**  
Autumn and spring terms.  
French or German up to A level standard.

**Comp. 161 Laboratory—programming and hardware**  
MR I. STEWART AND MR P.S. CUTLER  
Autumn, spring and summer terms.  
Practical work in support of material covered in the first year lecture courses.  
A collection of selected programming and hardware projects carried out using the computing facilities of the Department. The programming languages used are: Miranda, Modula–2, 68000 Assembler and Prolog. During the laboratory sessions students are given tuition to assist them with their individual work on the projects. Demonstrations of the equipment and operating systems to be used are also provided.
**Comp.162  Laboratory workshop I**  
MR I.D. STEWART AND MR P.S. CUTLER  
Autumn, spring and summer terms.  
Introduction to the projects of Comp.161 and the computer systems used to implement them.  
Presentation and discussion of the laboratory exercises. Instruction in the use of computing systems and associated computing facilities. Introductory descriptions of the relevant operating systems and language processors.

**Comp.163  Topics in AI (optional)**  
PROFESSOR D. GILLIES  
Spring and summer terms  
Introduction to some of state-of-the-art ideas, such as artificial intelligence, management, multimedia, parallelism, distributed systems, mathematical foundations, graphics and neural nets. Guided individual study and supporting lectures.

**Comp.164  Professional issues**  
MR J JOCOBSON  
Autumn term  
**Aims:** To introduce students to the organisational, ethical and legal contexts in which professional computing practitioners work.  
**Syllabus:** Professional Institutions: The role of professional institutions and their characteristics, the development and structure of the engineering profession, the origins and role of the Engineering Council, professional codes of conduct and codes of good practice (in particular those of the British Computer Society). Legal Framework for the Software Professional: Intellectual Property Rights—Law of Confidence, Copyright, Designs and Patents Act; the Thefts Acts and their limitations, Computer Misuse Act; Data Protection Acts; Plagiarism.

**Comp.176  Laboratory I (JMC)**  
MR P. CUTLER AND MR I. STEWART  
Autumn, spring and summer terms  
To support by practical work material covered in the first year lecture courses.  
**Computing**  
A collection of selected programming topics in Haskell, Turing, 68000 Assembler and Prolog carried out using the computing facilities of the department. Presentation and discussion of the laboratory exercises. Instruction in the use of computing systems and associated computing facilities. Introductory descriptions of the relevant operating systems and language processors.  
**Mathematics**  
Numerical quadrature: trapezoidal rule, elimination of successive orders of truncation error by Romberg extrapolation; Simpson's rule. Effect of singularities in the function or its derivatives upon accuracy/error estimates.  
Datapath and control unit: single-clock and multiple-clock implementations; microprogramming; exception handling. Memory hierarchy: caches, virtual memory. Pipelining: pipelined datapaths, data and branch hazards, exceptions. Advanced topics: hardware compilation, parallel architectures, special-purpose processors.
Comp. 202 Software engineering—algorithms
PROFESSOR A. WOLF AND PROFESSOR D. RUECKERT

Autumn term.

Aims: Expand your thinking about algorithms and algorithmic design.
Exposé you to several new classes of computational problems and concrete algorithmic solutions
Give you a sense for general (or commonly useful) approaches to algorithmic thinking

Learning outcomes: Students will be familiar with different algorithm design techniques
Students will be able to choose appropriate algorithm design techniques for particular algorithmic problems

Content: Review of asymptotic complexity analysis
Randomized algorithms: randomisation schemes; randomised binary search trees; skip lists; Bloom filters
String matching algorithms: brute force; Knuth-Morris-Pratt; Boyer-Moore
Radix searching algorithms: digital search trees; binary search trees; ternary search trees; suffix trees
Divide and conquer algorithms: Strassen’s matrix multiplication; Fibonacci numbers
Fast Fourier Transforms
Wavelet Transforms
Compression algorithms: lossless/lossy compression; Huffman; LZW; Wavelet/DCT compression
Dynamic programming
Greedy algorithms
Advanced graph algorithms: Flow networks; Maximum flow problems; Graph cuts; Ford-Fulkerson
algorithm; Edmonds-Karp algorithm
The course provides students with knowledge of several generally useful advanced algorithms. Topics
covered include: Randomised algorithms; String-matching algorithms; Dictionary search; Advanced graph
algorithms; Dynamic programming; Linear programming; Fourier transforms; Wavelet transforms.
There will be an examination and two assessed C++ programming exercises.

Comp.210 Computer architecture (optional)
PROFESSOR W. LUK

Spring term
To build on the foundation laid by modules on Computer systems and hardware; to show the relationship
between hardware and software; to focus on the concepts that provide the basis for current computers.
Introduction: overview; performance.
Instructions: formats, representations, interface with software. Arithmetic: number representation;
hardware for arithmetic operations; Arithmetic logic unit. Datapath and control unit: single-clock and
multiple-clock implementations; microprogramming; exception handling.
Memory hierarchy: caches, virtual memory.
Pipelining: pipelined datapaths, data and branch hazards, exceptions.
Advanced topics: hardware compilation, parallel architectures, special-purpose processors.

Comp.211 Operating systems II
DR P. PIETZUCH AND DR C. CADAR

Spring term.
Revision: real time programming and relation to OS vs definition and characteristics of real-time systems.
OS functions: resource management, providing a virtual machine. Non-determinacy.
Concurrent programming: concurrency in languages. Process interaction mechanisms. Semaphores,
Message passing.
The system kernel: hierarchical OS structure. Kernel functions and kernel entry. Process representation
and states. First level interrupt handler. Dispatcher and scheduling.
Implementation of semaphores, and message primitives.
Device management and input output: I/O software structuring. Device independent I/O.
Device and interrupt handlers. Spooler and buffering.
Time handling and scheduling: time handling facilities—delay and real-time clock. Scheduling strategies, priority pre-emption, and multilevel queues. Deadline specification and scheduling issues.


Security: access control and protection domains. Access control lists and capabilities.

Distributed operating systems: characteristics of distributed systems. Client server, pipeline and other interaction paradigms. Interaction primitives. Name servers and file servers.

**Comp.212 Networks and communications**

**DR P. MCBRIEN**

Autumn term.

Introduction and basic concepts: applications of computer communications and types of data, channels, bit rate and throughput network topologies, LANs, MANs and WANs, broadcast and point-to-point, synchronous and asynchronous communication. Connection oriented and connectionless communication.

Theoretical capacity of channels: Shannon’s law and the Nyquist relationship, multiplexing, queuing theory.

Computer communication system architectures: The OSI reference model: overview of the seven layer model. The TCP/IP model: comparison with the OSI model, overview of TCP/IP protocols.

The physical layer: transmission media. Properties of signals and signal degradation. Digital and analogue transmission, conversions between formats.


The medium access control sub-layer: slotted transmission, carrier sensing, token passing, distributed queues, IEEE MAC model and addresses, ethernet: 10Mbps to 1Gbps. Token ring, DQDB.

The network layer: switching; packet switched and circuit switched. Routing and internetworking.

Repeaters, bridges and routers flood routing, adaptive routing: link distance and backwards learning, internet protocols (IPv4, IPv6) and ARP. ATM, the public telephone network: POTS and ISDN.

The transport layer: UDP and TCP. Quality of service. The session layer.

The presentation layer: transport syntax: ASN.1 and ISO8825.

Security: confidential and authenticated data, public key and private key cryptography, digital signatures, PGP, key escrow, firewalls.

The application layer: OSI model protocols, DNS and URLs and FTP, HTTP, SMTP. Future directions: multimedia and the RTP.

**Comp.220 Software engineering—design I**

**DR E. LUPU, DR H.GUNES AND DR A. RUSSO**

Autumn term.

Principles and techniques for software design. Characteristics of software which motivate the need for software design; principles of good software design; different models of the software development process, with particular emphasis on the relationships between design and requirements, and design and implementation; and the role of modelling in design (e.g. object, dynamic and functional modelling) by applying techniques for specification and analysis (e.g. data-flow diagrams, state charts and state transition diagrams, object- and object-interaction diagrams, decision tables, use use cases, pseudo code).

**Comp.221 Compilers**

**DR N. DULAY AND PROFESSOR P.H.J. KELLY**

Autumn term.

Language processors, compilers, interpreters and their relatives. The structure of a compiler. Its context: editors, and loaders. Phases and passes. A complete example in two or three lectures.


**Comp.223 Concurrency (optional)**

DR S. UCHITEL AND PROFESSOR J. KRAMER

Spring term.

Introduction to concurrent programming: key ideas, interleaved actions, synchronisation, critical sections, deadlock, starvation, fairness, safety and likeness.

Monitors: entry queues, condition variables, wait and signal, alternative signalling mechanisms, reasoning about monitors.

Message passing: processes, synchronous and asynchronous communication, ports, send and receive, request-reply communication, non-deterministic choice, configuration programming.

Specification: introduction to formalisms for the specification and verification of concurrent system; labelled transition systems, process calculi CCS, CSP.

Software structure: structuring applications into modular, distributable software components; component types and instances; nesting and dynamic structures; component interfaces; connection patterns and naming.

**Comp.231 Introduction to Artificial Intelligence I (optional)**

DR F. TONI AND DR S. COLTON

Spring term.

Introduction to AI: the two streams of AI; Intelligence, some characteristics; classical AI examples; need for tests of success, the Turing test, ‘intelligence tests’, peer assessment; the potential and danger of the AI approach.

Problem solving: constraint satisfaction problems, generate and test methods with optimisations; state space problems. Search methods: local vs global information; heuristic search. Two person games: search and evaluation; adversary search; Minimax search; Alpha Beta search.

Expertise: characteristics of expert systems, separation of knowledge and inference, uniform representation, explanation; knowledge refinement; expertise and knowledge; accountability and the human window, human vs machine expertise.

Robotics: vision, scene recognition and the trihedral world, Waltz constraint satisfaction algorithm; speech synthesis, allophones, dictionary, morphology and rule based letter to sound approaches, stress and intonation.

Planning: viewed as state space search, means-ends heuristics; representation; separability and interactions, Sussman’s anomaly; example worlds, Blocks world, robot world and program synthesis; Problems and extensions, non-linear planning.

Learning: rote, trial and error, Pavlovian and history learning; Learning by exploration, hill climbing, linear games and programs, credit assignment and term selection problems; linear and non-linear evaluation functions; learning by search through operator and concept spaces.
Al and natural language: problem of understanding; Schank’s explanation game; Conceptual Dependency (CD) theory, primitive elements; Scripts, players, props, events, headers and exceptions; goal and plan directed understanding, role of themes.

**Comp.233  Computational techniques**

PROFESSOR A. EDALAT AND PROFESSOR P. HARRISON

Spring term.

Computation and accuracy.

Computational linear algebra: Vectors, matrices, special matrices, operations and algorithms. Vector and matrix norms. Vector spaces, linear dependence, bases, null space, rank space. Eigenvalues, Eigenvectors, singular values, singular value decomposition.


Basics of sparse computing.


Iterative methods.

Functions of several variables: Partial differentiation, the gradient, the Hessian. Taylor expansion.

Newton’s method for min f(x). Quadratic forms and linear systems. Method of conjugate gradients.

**Comp.240  Models of computation**

DR D. PATTINSON, DR N. PITERMAN AND DR S VAN BAKEL

Autumn term.

Evaluation and assessment methodology of Turing machine formalisation; construction and use of appropriate Turing machines. How to structure complex Turing machines from identified components; equivalence of generalisations of Turing machine; algorithmic unsolvability of certain problems; specified algorithms, rigorous correctness proofs and informal complexity analyses. Paradigm examples.

Complexity theory and P-time reduction. Non-deterministic Turing machines. Problems (TSP, HCP, etc.) solvable (non-deterministically) in polynomial time. The famous question P=NP of whether non-determinancy matters. NP-complete problems and Cook’s theorem on satisfiability of propositional formulas. Some useful graph algorithms.

**Comp. 245  Statistics**

DR N. HEARD

Autumn term.

The aim of this module is to equip students to make basic statistical analyses of data, and to enable them to critically assess and interpret others’ analyses. Detailed handouts were given during the course, as well as extensive problem sheets.

Introduction: What statistics is about: discovering structure in data. Why the popular misconception is wrong: modern statistics, greater statistics, statistics as the ultimate scientific instrument. Relationships between statistics and computer science through the overlapping areas of data mining, neural networks, machine learning, pattern recognition, etc. Examples of the sorts of projects my research students are working on. The space shuttle Challenger disaster as a motivating example. Examples of data from computer science (including probability, reliability, large databases, knowledge discovery, machine learning, etc.) Examples of questions which need statistical answers.

Simple graphical displays: Bar charts, histograms, pie charts, scattergrams, time series plots, contour plots, etc. How to (not to) cheat with graphs and how to detect when others are cheating.

Simple numerical summaries: Frequency distributions, stem and leaf plots.

Measures of location: mean, median, quartiles, mode

Measures of dispersion: range, variance (divide by n or n-1), standard deviation, interquartile range. Five figure summary, box plots skewness, tails of distributions.

Probability: Motivation: inference (drawing conclusions from samples in the context of natural variation in the population being studied), modelling (characterising the main features of the mechanism underlying
the process generating the data). Reminder of basic set theory, definition of probability Interpretations: classical, relative frequency, and subjective addition law of probability, independent events, multiplication law of probability, conditional probability, Bayes theorem.


**Central limit theorem.**

**Hypothesis tests:** Null hypothesis, alternative hypothesis, significance level, p-value, test statistic, rejection region (critical region). The two types of error: Type I: Rejecting a true null hypothesis Type II: Failing to reject a false null hypothesis Prob (Type I) is denoted ; Prob (Type II) is denoted The power of a test is . The steps in testing a hypothesis: 1: State null hypothesis, alternative hypothesis, significance level. 2: Select test statistic 3: Define rejection region 4: Do experiment 5: State conclusion 'Reject' or 'fail to reject' null hypothesis t-distribution, degrees of freedom. Tables of the t-distribution t-tests for comparing means with hypothesised values. One-sided and two-sided tests Two sample t-tests for comparing means of two populations Paired comparison (or matched pairs) tests for comparing means of two matched populations. Goodness-of-fit tests for categorical variables: observed values and expected values. The chi-square test statistic as an overall measure of discrepancy between observed values and the values one would expect if the hypothesised distribution was correct: , to be compared with a chi-squared distribution with d degrees of freedom, where d = k-p-1, where k is the number of categories and p is the number of parameters being estimated for the hypothesised distribution. Tables of the chi-squared distribution. Goodness-of-fit tests for continuous variables by grouping into categories and applying methods for discrete distributions. Chi-squared test for independence of two categorical variables: compute the expected values by multiplying the marginal probabilities (and rescaling by sample size). Then use standard chi-square test statistic above. Here, if there are r rows and c columns in the cross-classification, then the chi-squared distribution with (r-1)(c-1) degrees of freedom should be used.

**Estimation:** Population quantities: parameters. Sample quantities: statistics. Point estimates and interval estimates. An estimator - a formula, recipe, or algorithm for calculating the estimate from a sample. Different samples will yield different results - each of these is an estimate. Since estimates will vary from sample to sample, they will have a distribution - the sampling distribution. The bias of an estimator.


**Explanatory relationships:** Linear regression, and least squares estimation of the parameters. Correlation coefficient.

**Reliability theory:** Survivor (or reliability) function, failure rate function, hazard function. The special case of the exponential distribution, and its properties Series and parallel systems and their reliability.

**Computing:** Using Splus: a brief introduction, with hands on experience.

**Comp.252 Foreign language II (optional)**

Autumn and spring terms.

French or German language at post-A level standard.

**Comp.261 Laboratory II**

MR I. MOOR

Autumn, spring and summer terms.

Support by practical work of material covered in the second year lecture courses. A collection of projects on specification and design of software, language translators, operating systems, real-time control, databases, artificial intelligence and simulation. The programming languages used include C, C++, Prolog and Smalltalk.
Comp.262  Laboratory workshop II
MR I. MOOR
Autumn, spring and summer terms.
Introduction to the projects of Comp.261 and the computer systems used to implement them. Introduction and background to the practical exercises. Instruction in the use of computing systems and associated computing facilities. Introductory descriptions of the relevant operating systems and language processors.

Comp.302  Software engineering—methods
DR M. HUTH
Autumn term.
Testing: introduction to different stages of testing in the software lifecycle, focusing on unit testing applied to object-oriented programs. The main aim is to illustrate the integral development of tests with programs. Practical exercises using the JUnit test framework.
Design patterns: the rationale for design patterns. Introduction to their use by examination of a significant proportion of the gang-of-four patterns. Example programs and practical exercises. The overall objective is a better understanding of design for flexibility in non-trivial OO programs.
Ethical reasoning: ethical problems arising directly or indirectly in the context of IT professions. What makes such problems ‘ethical’? Common flawed reasoning in ethical arguments. ‘Considered moral judgements’.
Property rights: the history of philosophical justifications for property rights. Existing forms of protection of intellectual property: patents, copyright, trade secrets, trademarks, etc. Contemporary issues, such as Napster, the Microsoft anti-trust case and software patents.
Human values in computer design and technology: the potential social implications of computer technology and particular design decisions: bias in computer programs or designs; limited access (disabilities, internet connections, etc.); computer interfaces as social actors; and privacy.

Comp 303  Software engineering—systems verification
DR A. LOMUSCIO
Spring term.
Model checking packages: the nuSMV verification toolkit; MCMAS; SMV and ISPL syntax; lab classes; debugging; counter-models.
Verification scenarios: verification of PCI bus, verification of security protocols (anonymity and authentication).

Comp. 312 Advanced databases
DR P. MCBRIEN AND DR P. PIETZUCH
Autumn term.
Database management system architecture main components of a DBMS buffers, caches, and optimisation high level query languages and low level primitive operations. Concurrency control and recovery ACID properties of transactions, recoverability, serialisability. Transaction histories as a method for analysing database execution. Two-phase locking (2PL) ANSI SQL concurrency control levels. Query

**Comp. 317 Graphics**

PROFESSOR D. GILLIES AND PROFESSOR D. RUEKERT

Spring term.

This module covers the fundamental aspects of computer graphics that are utilised in the most prominent of its applications. The lectures cover:

- Device independent graphics: Raster and vector Devices, normalised device coordinates, world coordinates, the normalisation transformation, output primitives, input primitives.
- Raster graphics algorithms I: Line drawing, differential algorithms
- Raster graphics algorithms II: Filling, seed filling, dithering
- Planar polyhedra: three-dimensional data base representation, projection onto a viewing surface.
- Affine transformations and homogenous coordinates using projection.
- Clipping and containment in three-dimensional convex objects, splitting concave objects. 8) Texture mapping and anti-aliasing.
- Shading planar polygons: Gouraud shading, Phong shading.
- Polygon Rendering and Open GL
- Ray Tracing II: Computational efficiency, object space coherence, ray space coherence
- Radiosity 1: Modeling ambient light, form factors
- Radiosity 2: Specular effects, shooting patches, computational efficiency
- Geometric Warping
- Morphing Objects
- Special Visual Effects 1: Motion blur, lens models, caustics fog
- Special Visual Effects 2: Particle systems, fire, smoke and water

**Comp. 318 Custom computing**

PROFESSOR W. LUK

Overview: motivations; features and examples of custom computers; summary of development methods and tools.

Design: parametrised description of leaf components and composite structures; resource and performance characterisation; high-level design tools.

Optimisation: techniques for improving design efficiency such as pipelining, serialisation, transposition and their combinations.

Realisation: bit-level designs, data refinement, FPGA-based implementations.

System-on-chip: architectures, technology trade-offs, design and optimisation methods.

Examples will be selected from a number of application areas, including digital signal processing, computer arithmetic and non-numerical operations.
**Comp. 320 Complex systems**  
**PROFESSOR A. EDALAT**  
**Autumn term.**

In the course of the past decade complexity has become a major scientific field in its own right, which is sometimes considered the ultimate in interdisciplinary fields (EPSRC has named complexity among the five priority research areas along with energy, information communication technologies, nano-technology and healthcare). It relies on concepts in mathematics, physics and computer science on the one hand, and the rapid growth in computational power and information processing capabilities on the other.

This course aims to introduce students to this exciting new field, underlining its theoretical underpinnings, while engaging them in hands-on laboratory exercises in order to develop their practical skills and emphasise the applicability of this subject in the modern world.

**Comp. 332 Advanced computer architecture**  
**DR P.H.J. KELLY**  
**Spring term.**


**Comp. 333 Robotics**  
**DR A. DAVISON**  
**Spring term.**

How is independent locomotion achieved mechanically, electrically and controlled by software? What advantages are there to wheels and tracked systems? What additional problems are introduced by legged and ariel platforms? How does the robot know where it is in a local sense? The use of various sensors: infrared, sonar force sound and light sensors. The interfacing of such sensors to the microprocessor. Multiple robots need to communicate at a basic level before they can start to cooperate: radio and infrared links. The robot must be self-powered: batteries and efficient use of power, various techniques for economising on power. The reactive model for robot behaviour will provide a formal framework for most of this section but some details of the following areas will be discussed: control systems for each of the robot’s sub-functions, including open and closed loop systems; straight line control and wall following as contrasting examples. The problems of spatial orientation, recognition and environment interpretation. The use of a limited bandwidth communication channel. High-level planning and behaviour control software: arbitration schemes and behaviour-based architectures. Hybrid systems combining both deliberative and reactive architectures with knowledge
representation issues to produce interleaved planning and acting. Learning and adapting to changes in the environment, more abstract ideas of social behaviour, speech act communication and coordination schemes.

**Comp.335 Distributed systems**  
PROFESSOR M.S. SLOMAN  
Autumn term.  
Overview of distributed system architecture: motivation, system structures, architecture, ODP Reference model and distribution transparencies, design issues. Interaction primitives: message passing, remote procedure call, remote object invocation. Software structures and components: composite components, Darwin architecture description language, first and third party binding. Interaction implementation: message passing, RPC, concurrency and threads, heterogeneity of systems and languages. Security: threat analysis, security policies—military (Bell Lapadula) vs commercial models; access control concepts—identification, authentication, authorisation and delegation; authorisation policy: access matrix, access rules and domains, firewalls; access control lists, capabilities, secret and public key encryption, digital signatures, authentication, Kerberos; web security; security management. Distributed systems management: SNMP and OSI management models, monitoring and event generation, domains and policy.

**Comp.337 Simulation and modelling**  
DR A.J. FIELD, DR J. BRADLEY  
Autumn term.  
Introduction to system modelling using both computer simulation and mathematical techniques. Case studies with emphasis on the analysis of computer and communication systems using a combination discrete-event simulation and queueing theory.  
Introduction and basic simulation procedures.  
Model classification (with worked examples for each): Monte Carlo simulation, discrete-event simulation, continuous system simulation, mixed continuous/discrete-event simulation.  
Queueing networks: analytical and simulation modelling of queueing systems.  
Input and output analysis: random numbers, generating and analysing random numbers, sample generation, trace- and execution-driven simulation, point and interval estimation. Process-oriented and parallel simulation.

**Comp.341 Introduction to bioinformatics**  
DR M. VIGLIOTTI, PROFESSOR Y. GUO  
Spring term.  
Motivation from biology; scale of information being gathered; human genome project; drug design; potential of bioinformatics.  
Biological background—DNA; genes; protein structure, classification and prediction; expression concepts, drug design.  
Protein sequences—online databases; dot-plots; scoring schemes; BLAST algorithm; dynamic programming algorithm; structural inferences; profiles; PSI-BLAST; HMMs.  
Machine learning approaches—methodology; applications  
Inductive logic programming; neural networks; HMMs.  
Drug design—data mining; HT methods; analysis of gene, protein and metabolite expressions;  

**Comp.343 Operations research**  
PROFESSOR B. RUSTEM AND Dr D. KUHN  
Autumn term.  
Basic tools for quantitative methods for decision making. Solution methods and strategies. Underlying algorithms and result interpretation. Formulation of problems as abstract models which can be solved by generic algorithms.
Introduction to optimal decision-making processes in design and management. Necessary mathematical background and its application to solving a selection of constrained optimisation problems with special reference to computation.

Preview: optimal policy in design and management: mathematical models.

Linear programming: the Simplex method, two-phase Simplex method, duality, shadow prices.

Linear integer programming: Gomory's cutting plane methods for pure and mixed linear integer programming. Search methods; branch and bound algorithms.


**Comp. 345 Ludic computing**

This course builds on graphics, AI and HCI courses to equip the students with the technical and critical abilities required to build effective arts and entertainment software systems. The course aims to introduce key concepts in the design, construction and evaluation of interactive systems for play, including computer games and interactive art. There will be two main emphases: firstly, we will introduce key algorithms such as movement algorithms; path finding; rendering, decision making; and adaptation. Secondly, we will introduce key interaction design concepts and methods to enable the reflective design and assessment of arts/entertainment software, including models of user experience, and techniques for design, evaluation, data gathering and data analysis.

**Comp. 352 Humanities/foreign language/business**

A wide variety of subjects and languages are taught by staff from the Department of Humanities and the Business School.

See the Department of Humanities and Business School sections for details.

**BS0819 Organisations and management processes**

DR N. MALHOTRA

Spring term.

Introduction to basic theories of organisational behaviour and management. Introduction to techniques for evaluating the profitability of individual projects and for the construction and analysis of the main financial statements. Concepts of management organisation and behaviour; capital budgeting; analysis and interpretation of accounts.

Introduction to management: the evolution of managerial theories; managerial files and functions; the nature and type of managerial work.

Development of individuals and groups: issues of motivation, decision making and integration; dynamics of leadership; the learning organisation; human resources.

**Comp. 395 Machine learning**

PROFESSOR S. MUGGLETON AND DR M. PANTIC

This module aims at providing insights into advanced topics in artificial intelligence research and practice, concerning mostly machine learning techniques.

Introduction; Concept learning and the general-to-specific ordering; Decision tree learning; Artificial neural networks; Evaluating hypotheses; Computational learning theory; Instance-based learning; Genetic algorithms; Learning sets of rules; Analytical learning; Combining inductive and analytical learning; Reinforcement learning; Abductive reasoning and learning

**Comp. 401 Individual project**

Autumn, spring and summer terms

Individual project demonstrating independence and originality, the ability to plan and organise a large project over a long period, and to put into practice some of the techniques taught throughout the course.
Comp.417 Advanced graphics and visualisation
PROFESSOR D. RUECKERT, DR P. EDWARDS
Spring term.
Introduction to modern techniques in virtual reality, augmented reality and visualisation and their application to medicine and engineering.
Principals of visualisation: fundamentals and concepts, scalar, vector and tensor data.
Principles of visualisation: scalar, vector and tensor data, applications such flow visualisation.
Volume rendering: image-order rendering, object-order rendering. Scalar and ray transfer functions.
Isosurface generation, marching cubes algorithm.
Surface reconstruction 1: surface decimation, surface smoothing, surface normal generation.
Surface reconstruction 2: surface triangulation, Voronoi diagrams, Delaunay triangulation.
Spline curves 1: parametric and non-parametric splines, cubic spline patches.
Spline curves 2: Bezier curves, B-spline formulation.
Surface modelling 1: Coons patches, bi-cubic surfaces.
Surface modelling 2: B-spline surfaces, NURBS, rendering spline surfaces.
Implicit surface models, soft objects.
Image-based rendering and lightfields.
Virtual reality: stereo perception, stereo display, head-mounted displays, autostereoscopic displays, holographic displays. Haptic and tactile force feedback. Virtual worlds, collision detection for VR.
Augmented reality: definitions and examples. Calibration and tracking.
Simulation training in medicine 1 and 2.

Comp.418 Computer vision
DR G.Z. YANG
Autumn term.
Introduction to the concepts behind computer-based recognition and extraction of features from raster images. Illustration of some successful applications of vision systems and their limitations
Overview of early, intermediate and high level vision: first and second moments of illumination for recognition and classification of machine shop components in silhouette.
Segmentation: region splitting and merging; quadtree structures for segmentation; mean and variance pyramids; computing the first and second derivatives of images using the isotropic, Sobel and Laplacian operators; grouping edge points into straight lines by means of the Hough transform; limitations of the Hough transform; parameterisation of conic sections.
Perceptual grouping: failure of the Hough transform; perceptual criteria; improved Hough transform with perceptual features; grouping line segments into curves.
Overview of mammalian vision: experimental results of Hubel and Weisel; analogy to edge point detection and Hough transform; neural networks based on the mammalian vision system.
Relaxation labelling of images: detection of image features; simulated annealing.
Grouping of contours and straight lines into higher order features such as vertices and facets.
Depth measurement in images; triangulation; projected grid methods; shape from shading based on multisource illumination.
Matching of images: the correspondence problem for stereo vision; two camera and multiple camera systems; shape from motion as a further stage of stereo vision; optical flow between adjacent video frames.
Expert system modelling in computer vision: model based vision using inference engines and rules.
**Comp.420 Cognitive robotics**
PROFESSOR M. SHANAHAN
Spring term.
Introduction
Event calculus
Planning through abduction
Perception through abduction
Interleaving perception, planning, and action
Hierarchical planning
Situation calculus
The Golog language
Brain-inspired architectures
Internally closed sensorimotor loops
The global workspace architecture

**Comp.422 Computational finance**
DR K. KUHN
Aims: The purpose of this course is to study the basic tools of Computational Finance. This entails the study of computational models and quantitative methods.
Learning outcomes: The student will be able to perform the basic computational functions related to financial engineering.
Syllabus: The course will discuss computational aspects of financial engineering. This includes computational models, algorithms and software design.

**Comp. 424 Machine learning and neural computation**
DR A. FAISAL
Spring term.
The course will cover foundations and methods of it’s interrelated title-topic and be related to our current understanding in neuroscience of how the brain and our sensor motor system solves these problems. The course will cover basic foundations (Bayes probability theory, decision theory, optimal control). The course will use these to address topics related to learning, decision making and motor control. More specifically we will address unsupervised learning, supervised learning (function approximation and interpolation/extrapolation), dimensionality reduction, classification and decision making and reinforcement learning (as a way to understand how dynamic behaviour in a dynamic world is learned). I would teach this course with examples or motivations how these methods are founded in actual examples of how we (our brain) infers or makes decision about the world. An important concept to me is to give students a first-principle quantitative introduction in the foundations part and then show how the above methods work and why they are statistically optimal and what the assumptions behind these models are. This course thus supplies, beyond the relevance for neuroscience, applicable knowledge to the areas of data mining, robotics, computational finance and operations research. It integrates, without overlapping with the Machine Learning and the Cognitive Robotics course offered in Computing and the Computational Neuroscience course in Biomedical Engineering.
Comp.429 Parallel algorithms
PROFESSOR P.G. HARRISON AND DR N. DINGLE
Spring term.
Introduction and motivation: key concepts, performance metrics, scalability and overheads.
Classification of algorithms, architectures and applications: searching, divide and conquer, data parallel.
Static and dynamic, message passing and shared memory, systolic.
Sorting and searching algorithms: mergesort, quicksort and bitonic sort, implementation on different architectures. Parallel depth-first and breadth-first search techniques.
Matrix algorithms: striping and partitioning, matrix multiplication, linear equations, eigenvalues, dense and sparse techniques, finite element and conjugate gradient methods.
Optimisation: graph problems, shortest path and spanning trees. Dynamic programming, knapsack problems, scheduling, element methods.
Synthesis of parallel algorithms: algebraic methods, pipelines, homomorphisms.

Comp.430 Network security
DR M. HUTH AND DR E.C. LUPU
Spring term.
Survey of principles and practice of network security. Underlying principles and techniques of network security with examples of how they are applied in practice.
Introduction: assets, threats, countermeasures; network security models, security functions: confidentiality, authentication, integrity, nonrepudiation, access control, availability, passive and active attacks, end-to-end vs link-to-link encryption.
Classical cryptography: key ideas, steganography, codes, one-time pad, substitution and transposition ciphers, cryptanalysis, cryptographic strength.
Symmetric-key cryptography: Feistel cipher; DES: basics, rounds, e-box, s-box, p-box, key box; Modes of operation: ECB, CBC, CFB, OFB; double DES, triple DES, IDEA, RC5, AES, problems with symmetric key cryptography.
Public-key cryptography: requirements, confidentiality, authentication, modular arithmetic, Diffie-Hellman key exchange, RSA, attacks against RSA, hybrid cryptosystems, elliptical curve, quantum cryptography.
Digital signatures: characteristics, MACs, one-way hash functions, signing and verification, birthday attack, public-key certificates, disavowed signatures, arbitrated digital signatures, chaffing and winnowing.
Key management: distribution, KDC, announcements and directories, public key certificates, X509 certification authorities, PGP web of trust, control vectors, key generation and destruction, key backup.
Intruders and programmed threats: host access, password systems and attacks, one-time passwords, token cards, biometrics, trapdoors, programmed threats: trapdoors, logic bombs, trojan horses, viruses, worms, countermeasures, intrusion-direction.
Firewalls: internet security policies, firewall design goals, firewall controls, TCP/IP, packet filtering routers, application-level gateways, circuit-level gateways, firewall architectures, VPNs.
Web security: WWW, web servers, CGI, active content, Java applets, Java security model: sandbox, class loaders, bytecode verification, security manager, Java attacks, bypassing Java, mobile code cryptography.

Comp.436 Performance analysis
PROFESSOR P.G. HARRISON, DR J. BRADLEY
Spring term.
Introduction to analytical modelling techniques for predicting computer system performance.
Motivation and survey; the need for performance prediction in optimisation and system design.
Basic probability theory: renewal processes; Markov processes; birth and death processes; the single server queue; Little’s law; embedded Markov chain; M/G/1 queue; queues with priorities; queueing networks—open, closed, multi-class; equilibrium state space probabilities, proof for single class; normalising constants; computation of performance measures; convolution algorithm; mean value analysis; application to multi-access systems with thrashing.

Decomposition and aggregation: Norton’s theorem; M/M/n queue; multiple independent parallel servers.

**Comp.437 Distributed algorithms**
PROFESSOR A. WOLF
Spring term.
Models of distributed computing
Synchrony, communication and failure concerns
Synchronous message-passing distributed systems
Algorithms in systems with no failures—Leader Election and Breadth-First Search algorithms
The atomic commit problem
Consensus problems—the Byzantine General’s Problem
Asynchronous message-passing distributed systems
Logical time and global system snapshots
Impossibility of consensus
Fault-tolerant broadcasts
Partially synchronous message-passing distributed systems
Failure detectors
The Labelled Transition System Analyser (LTSA) tool www.doc.ic.ac.uk/~jnm/book/ltsa-v2/index.html is used throughout the course for modelling and demonstrating the execution of various algorithms.

**Comp.438 Complexity**
DR I.C.C. PHILLIPS
Spring term.
Description of the complexity classes associated with computational problems. How to fit a particular problem into a class of related problems, and so to appreciate the efficiency attainable by algorithms to solve the particular problem.
Turing machines, decidability, machine independence.
Time complexity: the classes P and NP, NP-completeness, example problems from logic and graphs. Space complexity classes. The parallel computation thesis, PRAMs, the class NC. Probabilistic algorithms.

**Comp.452 Humanities/foreign language/business**
A wide variety of subjects and languages are taught by staff from the Department of Humanities and the Business School. See the Department of Humanities and the Business School syllabuses for details.

**Comp.461 Project outsourcing exercise**
Autumn term.
Practical support for the compulsory and optional modules. A substantial group project with an emphasis on maintenance forms the major component of this module.

**Comp.464 Industrial placement—presentation and report**
MS A. ALLINSON
Autumn term.
Verbal and written communication skills required in the practice of software engineering. Integration of industrial placement with academic studies. Preparation of short seminar on a technical subject related to the industrial placement and submission of a log book describing the training and responsibilities during the placement.
Comp 470 Program analysis
DR H. WIKLICKY AND PROFESSOR C. HANKIN
Spring term.
Aims: To gain an understanding of program analysis techniques that are used in optimising compilers for imperative and functional programming languages. To develop understanding of data flow analysis techniques, control flow analysis techniques and constraint solving.
Learning outcomes: The student who completes this module will be able to apply classical analyses to programs; solve the resulting constraints; develop analyses from scratch for both imperative and functional programming languages; and informally reason about the complexity of constraint solving algorithms.

Comp 471 Advanced issues in object-oriented programming
PROFESSOR S. DROSSOPOULOU
Autumn term.
Issues around the design and implementation of object oriented languages, their rationale and explore alternatives. Formal calculi as an unambiguous notation, and as a way to establish soundness. Motivation (problems with the Eiffel type system, C++ type system). Class based vs object based languages. Outline of implementation of three characteristic oo languages (Smalltalk, Java, C++) oo 1: a calculus for a simple, imperative oo language without inheritance; the representation of objects in Java/C++. oo 2= oo 1 with types; well-formed programs, subsumption, subject reduction theorem, meaning and proof. Overloading vs multimethods: rationale and implementation issues. oo 3a= oo 2+ overloading; oo 3b= oo 2+ multimethods. Class hierarchies, rationale for inheritance; oo 4= oo 2+ inheritance, Subject reduction theorem and proof. oo 5+ oo 4+ overloading, Subject reduction theorem and proof; method calls in C++ in the presence of overloading. Stack vs heap objects: dynamic vs static binding in C++; aliases and related problems; flexible alias protection. Multiple supertypes: multiple inheritance in C++ rationale, implementation; multiple superinterfaces in Java, rationale, implementation. The Java implementation architecture, binary compatibility. Alternative calculi: the Abadi-Cardelli calculus, the Mitchell-Fischer calculus. Parameterised classes and oo programming: C++ templates, GJ, mixins, mixed Java. Object based languages, programming in SELF.

Comp 474 Multi-agent systems
DR F. TONI
Spring term
Comp.475  Advanced topics in software engineering  
PROFESSOR S. EISENBACK DR N. PRYCE AND MR A. MCVEIGH  
Autumn term.  
Software process development techniques: current methods for developing systems, i.e. extreme programming, use of open source software, web development methodologies and component-based programming. How to choose the most suitable methodology for each task and evaluate new methodologies as they become popular.  
Maintenance: modification of a software product after delivery to correct faults to improve performance or other attributes, or to adapt the product to a changed environment. Software maintenance problems associated with old and new code (Fortran and Java). Reverse engineering techniques for code improvement.  
Object modelling notations: formal object modelling languages, especially Daniel Jackson’s Alloy (MIT). Use of Alloy and its automated tool, Alcoa, to validate or refute practical object models. Comparison of Alloy to two other object modelling languages: UML and Z.  
Software verification: importance of formal and automatically verifiable lightweight specifications of program behaviour for successful programming in the large. Presentation and application of either a behavioural interface specification language for Java (JML) or Compaq’s extended static checking tool for Java (ESC/JAVA) as a representative of the state-of-the-art in this strategically important area.  
Requirement engineering and consistency checking: main objectives. Place of requirement engineering in the software life cycle. Effective requirement practices. Inconsistency management in software requirements and specifications. Presentation and evaluation of the tool xlinkit (UCL) which checks for consistencies of distributed XML documents with respect to specifications written in first order logic.  

Comp.477  Computing for optimal decisions  
PROFESSOR B. RUSTEM  
Autumn term.  
Optimal decision making models, algorithms and applications to finance. Algorithm design and formulation of decision models. Analysis of algorithms and interpretation of results. Issues related to current research.  
Mathematical concepts and advanced computational methods for quantitative problems in management decision making. Unconstrained and constrained optimal decision formulations and associated optimality conditions. Quadratic and general non-linear programming formulations and algorithms.  

Comp.480  Automated reasoning  
DR K. BRODA  
Propositional and first order automated reasoning methods (Davis Putnam, Model Generation, resolution and refinements, connection graphs, Tableau methods, paramodulation and equational reasoning) and be able to carry out small examples by hand. Soundness and completeness of the techniques. Otter theorem prover to solve problems and to investigate search spaces. Knuth Bendix completion procedure and its applications to equational reasoning.  
To describe the main techniques for automated reasoning in classical logic.  
Refinements of resolution, hyper-resolution, locking, connection graphs, advantages, disadvantages, use of the Otter Theorem prover.
Tableau methods with unification; linear refinements, extensions.
Paramodulation, theory resolution.
Equational methods for equational theories.
Confluent rewriting systems and the Knuth-Bendix algorithm, unifying Knuth-Bendix.

Comp.481 Models of concurrent computation
PROFESSOR P GARDNER AND DR N. YOSHIDAAutumn term
An overview of various formalisms for the specification and verification of concurrent systems.
Overview: communication via message passing or shared variables, non-determinism, processes as ‘black boxes’, interleaving versus true concurrency.
CCS (calculus of concurrent systems): operators, case studies, concurrency workbench (software tool), transition systems, equational reasoning, bisimulation, observational congruence, Hennessy-Milner logic, timed CCS.
Pi calculus and related calculi, e.g. Spi calculus for cryptographic protocols.
CSP (communicating sequential processes): operators, case studies, failures model, relationship with CCS. True concurrency: event structures, Petri nets, case studies, relationship with CCS/CSP.

Comp.482 Type systems for programming languages
DR S VAN BAKEL
Autumn term.
Lambda Calculus: terms, abstraction, application, reduction, normal form, normalisation, head normal form, head normalisation. The Curry type assignment System.
Types, type assignment rules, type substitution, unification, subject reduction. The Principle type for Curry's System.
Recursion and Polymorphism: the need for a recursor, how to type recursion, Milner’s let, the basic ML system, the principal type property for the ML system. Church’s typed lambda calculus: type checking versus type inference.
The Polymorphic Lambda calculus. Strong normalisation for the polymorphic Lambda Calculus.
Intersection type assignment: types, type assignment rules, subject expansion, undecidability, filter semantics.
Patterns: term rewriting systems, weak reduction, normal forms, strong normalisation. Dealing with patterns, recursion, subject expansion and reduction. A decidable restriction of the intersection system.

Comp.484 Quantum computing
DR H. WIKILICKY
Spring term.
Introduction to quantum mechanics, quantum bits and complex vector spaces, quantum evolution and quantum gates, quantum registers, universal gates, no-cloning theorem, quantum entanglement and teleportation, quantum algorithms, quantum search, quantum fourier transform, phase estimation, quantum counting, order finding for periodic functions, quantum factoring of integers, physical realization of quantum gates and quantum error correction.

Comp.491 Knowledge representation
PROFESSOR M. SERGOT
Spring term.
The theoretical foundations of the main formalisms for knowledge representation and reasoning, how they can be used in practice, and an overview of current research trends. Logic-based formalisms, and comparison and translation between different approaches.
Current general issues in knowledge representation. Default and non-monotonic reasoning; default logic, autoepistemic logic, circumscription, negation as failure, abduction as a form of hypothetical and default
reasoning, non-monotonic consequence relations and defeasibility. Theories of argumentation. Temporal reasoning; representation of action, non-monotonic features of persistence, the frame problem. Dynamics of belief systems and databases: consistency and integrity, knowledge assimilation, theories of belief revision and update, counterfactuals.

**Comp.493 Intelligent data and probabilistic inference**  
PROFESSOR D.F. GILLIES  
Autumn term.  
Foundation of intelligent data analysis: intelligent data analysis, data mining and Bayesian methods. Classification, clustering and causality analysis. Statistic methods: sampling, statistical inference. Induction methods and classification, overfitting and model evaluation.  
Tree/rule induction.  

**Comp.499 Modal and temporal logic**  
PROFESSOR I HODKINSON DR C. KUPKE AND DR N. BEZHANISHVILI  
Autumn term.  

**Comp.526 Databases**  
MR J. JOCOSON  
Spring term.  
A general introduction to database systems. The theory and practice of the relational model in detail, including design, database languages, transaction management and recovery and concurrency. Entity-relationship modelling and translation to the relational model. Introduction to databases, including data modelling, database management, data dictionary, query formulation and evaluation. Relational databases: design, functional dependencies, normalisation up to and including the fourth normal form, losslessness, dependency preservation.
Relational database languages, including relational algebra and relational tuple calculus. Views and database integrity.
Transaction management and recovery: transaction atomicity, database log, commit and rollback, recovery from system and media failure, deferred and immediate database modifications, checkpoints.
Concurrency: including conflict serialisability, conflict equivalence, precedence graphs, serialisability, locking, two-phase locking protocol, deadlock.
Entity-relationship modelling and translation to the relational model.
The module is supported by laboratory sessions using the relational database system, INGRES, and the language SQL.

Comp.527 Computer networks and distributed systems
PROFESSOR M. SLOMAN AND DR P. PIETZUCH
Spring term.
Network overview: interfaces, protocols and services, connection-oriented and connectionless services, OSI and TCP/IP reference models.
Local area networks: topologies—star, bus, ring, media access control (deterministic and probabilistic), IEEE 802.x. wireless networking.
Data link protocols: framing and data transparency, error detection and correction, flow control.
Interconnecting networks: transparent and source routing bridges, switches, routers - adaptive and non-adaptive routing protocols.
Internet protocols: IP Addressing, ARP and RARP, IP and ICMP, UDP and TCP.
Overview of distributed system architecture: motivation, system structures, ODP reference model and distribution transparencies, design issues.
Interaction primitives: message passing, remote procedure call, remote object invocation.
Interaction implementation: message passing, RPC, concurrency and threads
Security: threat analysis, security policies—military (Bell Lapadula) versus commercial models; access control concepts—identification, authentication, authorisation and delegation; authorisation policy—access matrix, access rules and domains; access control lists, capabilities, secret and public key encryption, digital signatures, authentication, Kerberos.