Faculty of Science

BERN09, Computational Science: Parallel Programming in Scientific Computing, 7.5 credits

Beräkningsvetenskap: Parallellprogrammering inom beräkningsmatematik, 7,5 högskolepoäng

Second Cycle / Avancerad nivå

Details of approval

The syllabus was approved by The Education Board of Faculty of Science on 2025-06-13. The syllabus comes into effect 2025-06-13 and is valid from the spring semester 2026.

General information

The course is an alternative-compulsory course for a degree of Master of Science in Computational Science and an elective course for a Degree of Master of Science in mathematics with a specialisation in numerical analysis. The course can also be given as an independent course.

Language of instruction: English

Main field of study	*Specialisation*
Computational Science	A1F, Second cycle, has second-cycle course/s as entry requirements
Mathematics with specialization in Numerical Analysis	A1F, Second cycle, has second-cycle course/s as entry requirements

Learning outcomes

Parallel programming is a widely deployed programming technique to improve the computational speed of numerical algorithms. Depending on the nature and size of the computational problem at hand, a speed up of several orders of magnitude is achievable.

The course aims to provide an understanding of the key concepts of parallel programming and to make students aware of potential problems affecting program correctness and hindering the computational speed. On completion of the course students should have in-depth knowledge to successfully develop, modify, extend and

assess parallel algorithms and parallel programs.

Knowledge and understanding

On completion of the course, students shall be able to:

- describe key concepts in shared and distributed memory parallel programming
- exemplify common performance inhibitors in parallel computing
- exemplify possible causes for parallel programs stopping execution or delivering non-reproducible results.

Competence and skills

On completion of the course, students shall be able to:

- write parallel programs and parallelise existing programs to improve the computational speed
- write, extend and maintain shared memory parallel programs using OpenMP directives
- write, extend and maintain message passing programs using the MPI Application Programming Interface (API).

Judgement and approach

On completion of the course, students shall be able to:

- explain advantages and disadvantage of the distributed and shared memory programming models
- assess the performance of parallel algorithms and parallel programs
- make an informed choice between shared memory and distributed memory or whether there is a benefit from deploying multiple programming models simultaneously.

Course content

The course treats:

- Introduction to programming in C
- Concepts behind distributed and shared memory parallel computing
- Commonly used commands of the MPI and OpenMP® API
- Performance assessment of parallel applications
- Debugging and profiling of parallel codes
- Efficient OpenMP and MPI parallelisation of standard algorithmic constructs such as matrix-vector-products, scalar products, unbalanced summations, recursive calls, divide and conquer.
- Hybrid parallelisation (shared and distributed memory) of a stencil code on a regular grid.

Course design

Teaching consists of lectures, tutorials and computer exercises. A compulsory programming project is included in the course.

Assessment

The examination is conducted in writing through assignments submitted during the course and through a written report of the programming project. The examination also includes oral explanations of the assignments as well as a presentation of the programming project at the end of the course.

For students who did not pass the regular examination, an additional opportunity will be offered during scheduled re-examination period.

The examiner, in consultation with Disability Support Services, may deviate from the regular form of examination in order to provide a permanently disabled student with a form of examination equivalent to that of a student without a disability.

Grades

Grading scale includes the grades: Fail, Pass, Pass with distinction
The grading scale for the assignments is Fail, Pass, while the written report and the oral presentation of the project, that also includes questions regarding the assignments, is assessed according to the grading scale Fail, Pass, Pass with Distinction.

To pass the whole course, the student must have passed the assignments, the written report, and the oral presentation of the project, including the questions about the assignments. The final grade is determined by the grade on the project.

Entry requirements

For admission to the course English 6/B and at least 90 credits in natural science are required, including knowledge equivalent to the course NUMA01 Computational Programming with Python (7.5 credits). In addition, knowledge equivalent to one of the following courses is required: NUMA32 Numerical Methods for Differential Equations (7.5 credits), NUMN21 Advanced Course in Numerical Algorithms with Python/SciPy (7.5 credits), FYTN03 Computational Physics (7.5 credits) or FYSN33 Applied Computational Physics and Machine Learning (7.5 credits).

Further information

This course is also given in the third cycle, course code NTF010F.

The course may not be credited together with NTF010F Parallel programming of HPC (High Performance Computing) systems.

The course is offered by the Centre for Mathematical Sciences, Lund University.