Computational Physics Lectures: Introduction to the course

Morten Hjorth-Jensen[1,2]

[1]Department of Physics, University of Oslo
[2]Department of Physics and Astronomy and National Superconducting Cyclotron Laboratory, Michigan State University

Aug 19, 2020

Overview of first week

- Thursday: First lecture: Presentation of the course, aims and content
- Thursday: Second Lecture: Introduction to C++ programming (chapters 2 and 3 of lecture notes) and start discussion of project 1.
- Friday: Numerical precision and C++ programming, continued and discussion of project 1 (chapter 2 and 3 of lecture notes)
- Numerical differentiation and loss of numerical precision (chapter 3 lecture notes)
- Computer lab: Thursday and Friday. First time: Thursday and Friday this week, Presentation of hardware and software The first two weeks we focus on simple programming exercises, start with project 1 and to set up GitHub/GitLab (with **git** as version control software) and QTcreator as IDE (integrated Development Environment). This week we discuss how to set up git and obtain a GitHub/GitLab account and look at simple programming exercises and for those interested start with project 1. We discuss also how to install C++ compilers.
- The recommended programming exercises for the first week: Exercises 3.1, 2.1 and 2.2 in Lecture notes. **Exercise 3.1 is the most relevant for project 1.**

Reading suggestions and exercises

- Introduction to C++ programming
- Numerical precision and C++ programming (chapter 2 of lecture notes)
- Numerical differentiation and loss of numerical precision (chapter 3 lecture notes)

Lectures and ComputerLab

- Lectures: Thursday (8.15am-10am) and Friday (8.15am-10am), Store Fys Aud, but due to Covid-19 all lectures, unless announced, will be fully online and in a flipped style.
- Weekly reading assignments needed to solve projects. This includes written material and videos as well
- First hour of each lab session may be used to discuss technicalities, address questions etc linked with projects.
- Detailed lecture notes, exercises, all programs presented, projects etc can be found at the homepage of the course.
- Computerlab: Thursday (10am-6pm, room FV203, Dept of Physics) and Friday (10am-6pm, room FØ434, Dept of Physics). Depending on the number of participants, we may extend to more lab sessions. Please bring your own laptops to the lab sessions.
- Weekly plans and all other information are on the official webpage.
- No final exam, the last three projects are graded. In total five projects which all have to be approved.

Course Format

- Five compulsory projects. Electronic reports only using devilry to hand in projects and Git for repository and all your material.
- Evaluation and grading: The last three projects are graded and each counts 1/3 of the final mark. No final written or oral exam.

- C/C++ is the default programming language during lectures, but Fortran and Python are also used. All source codes discussed during the lectures can be found at the webpage and github address of the course. We recommend either C/C++, Fortran or Python as programming languages. You can also try Julia or other languages but at the Lab and during the lectures we will focus on the first languages.

Teachers and ComputerLab

Teachers:

- Morten Hjorth-Jensen
  - **Email**: morten.hjorth-jensen@fys.uio.no
  - **Phone**: +47-48257387
  - **Office**: Department of Physics, University of Oslo, Eastern wing room 470
  - **Office hours**: *Anytime*! In Fall Semester 2020 (FS20), as a rule of thumb office hours are planned via computer or telephone. Individual or group office hours will be performed via zoom. Feel free to send an email for planning. In person meetings may also be possible if allowed by the University of Oslo's COVID-19 instructions.
- Anders Kvellestad
  - **Email**: anders.kvellestad@fys.uio.no
  - **Office**: Department of Physics, University of Oslo, Eastern wing room 447

Teaching Assistants FS20:

- Sebastian G. Wither-Larsen, s.g.winther-larsen@fys.uio.no
  - **Office**: Department of Physics, University of Oslo, Eastern wing room 454
- René Alexander Ask, r.a.ask@fys.uio.no
- Kaspara Gåsvær, k.s.gasvar@fys.uio.no
- Maria Linea Horgen, m.l.horgen@fys.uio.no
- Aksel Graneng, akselgraneng@gmail.com

Thursday	Friday
Group 1: 10am-12pm	Group 5: 10am-12pm
Group 2: 12pm-2pm	Group 6: 12pm-2pm
Group 3: 2pm-4pm	Group 7: 2pm-4pm
Group 4: 4pm-6pm	Group 8: 4pm-6pm

Groups are in person but we are also planning fully online groups if needed.

Deadlines for projects (end of day, tentative deadlines)

1. Project 1: September 9 (not graded, only feedback)
2. Project 2: September 30 (not graded, only feedback)
3. Project 3: October 21 (graded with feedback)
4. Project 4: November 18 (graded with feedback)
5. Project 5: December 9 (graded with feedback)

Projects are handed in using **canvas**. We use Github or GitLab (eventually Bitbucket) as repository for codes, benchmark calculations etc. Comments and feedback on projects only via **canvas**.

Topics covered in this course

- Numerical precision and intro to C++ programming
- Numerical derivation and integration
- Random numbers and Monte Carlo integration
- Monte Carlo methods in statistical physics
- Quantum Monte Carlo methods
- Linear algebra and eigenvalue problems
- Non-linear equations and roots of polynomials
- Ordinary differential equations
- Partial differential equations
- Parallelization of codes
- High-performance computing aspects and optimization of codes

Syllabus

Linear algebra and eigenvalue problems, chapters 6 and 7.

- Know Gaussian elimination and LU decomposition
- How to solve linear equations
- How to obtain the inverse and the determinant of a real symmetric matrix
- Cholesky and tridiagonal matrix decomposition

Syllabus

Linear algebra and eigenvalue problems, chapters 6 and 7.

- Householder's tridiagonalization technique and finding eigenvalues based on this
- Jacobi's method for finding eigenvalues
- Singular value decomposition
- Cubic Spline interpolation

Syllabus

Numerical integration, standard methods and Monte Carlo methods (chapters 4 and 11).

- Trapezoidal, rectangle and Simpson's rules
- Gaussian quadrature, emphasis on Legendre polynomials, but you need to know about other polynomials as well.
- Brute force Monte Carlo integration
- Random numbers (simplest algo, ran0) and probability distribution functions, expectation values
- Improved Monte Carlo integration and importance sampling.

Syllabus

Monte Carlo methods in physics (chapters 12, 13, and 14).

- Random walks and Markov chains and relation with diffusion equation
- Metropolis algorithm, detailed balance and ergodicity
- Simple spin systems and phase transitions
- Variational Monte Carlo
- How to construct trial wave functions for quantum systems

Syllabus

Ordinary differential equations (chapters 8 and 9).

- Euler's method and improved Euler's method, truncation errors
- Runge Kutta methods, 2nd and 4th order, truncation errors
- How to implement a second-order differential equation, both linear and non-linear. How to make your equations dimensionless.
- Boundary value problems, shooting and matching method (chap 9).

Syllabus

Partial differential equations, chapter 10.

- Set up diffusion, Poisson and wave equations up to 2 spatial dimensions and time
- Set up the mathematical model and algorithms for these equations, with boundary and initial conditions. Their stability conditions.
- Explicit, implicit and Crank-Nicolson schemes, and how to solve them. Remember that they result in triangular matrices.
- How to compute the Laplacian in Poisson's equation.
- How to solve the wave equation in one and two dimensions.

Overarching aims of this course

- Develop a critical approach to all steps in a project, which methods are most relevant, which natural laws and physical processes are important. Sort out initial conditions and boundary conditions etc.
- This means to teach you structured scientific computing, learn to structure a project.
- A critical understanding of central mathematical algorithms and methods from numerical analysis. In particular their limits and stability criteria.
- Always try to find good checks of your codes (like solutions on closed form)
- To enable you to develop a critical view on the mathematical model and the physics.

Additional learning outcomes

- has a thorough understanding of how computing is used to solve scientific problems
- knows some central algorithms used in science
- has knowledge of high-performance computing elements: memory usage, vectorization and parallel algorithms
- understands approximation errors and what can go wrong with algorithms
- has experience with programming in a compiled language (Fortran, C, C++)
- has experience with debugging software
- has experience with test frameworks and procedures
- can critically evaluate results and errors
- understands how to increase the efficiency of numerical algorithms and pertinent software
- understands tools to make science reproducible and has a sound ethical approach to scientific problems
- Is able to write a scientific report with software like Latex

Computing knowledge

Our ideal about knowledge on computational science

Hopefully this is not what you will feel towards the end of the semester!

©Zits Partnership

And, there is nothing like a code which gives correct results!!

Other courses in Computational Science at UiO

Bachelor/Master/PhD Courses.

- FYS-STK3155/4155 Applied data analysis and machine learning, fall semester
- FYS4411 Computational physics II (Parallelization (MPI), object orientation, quantum mechanical systems with many interacting particles), spring semester
- FYS4460 Computational physics III (Parallelization (MPI), object orientation, classical statistical physics, simulation of phase transitions, spring semester
- MAT4110 Introduction to Numerical Analysis
- MAT3360 - Introduction to Partial Differential Equations
- IN5270 Numerical methods for PDEs, finite element method
- IN3110 Problem solving with high-level languages (Python), fall semester
- IN3200/4200 Parallel computing for problems in the Natural Sciences (mostly PDEs), spring semester

Extremely useful tools, strongly recommended

and discussed at the lab sessions.

- GIT for version control (see webpage), this week at lab
- ipython notebook

- QTcreator for editing and mastering computational projects (for C++ codes, see webpage of course), discussed during second week
- Armadillo or Eigen as a useful numerical libraries for C++, highly recommended, week 36 and rest of semester
- Unit tests, week 37 and later