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

2016

Overview of first two weeks

- First lecture: Presentation of the course, aims and content. Introduction to github with tutorial.
- Second week (read sections 2.1-2.5 and 3.1-3.2 of lecture notes):
  - 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)
  - Work on warm up exercise to demonstrate several programming elements

Lectures and ComputerLab

- Lectures: 1-3pm Wednesdays (BPS4270) and 7-9pm Wednesdays (BPS1300).
- Lab: 2-4pm Thursdays and 1-3pm Fridays in BPS1240.
- Weekly reading assignments needed to solve projects.
- First part of a lab session may be used to discuss technicalities and issues related to additional software

- Detailed lecture notes, exercises, all programs presented, projects etc can be found at the homepage of the course.
- Weekly plans and all other information are at the github address of the course. Weekly emails about plans and progress will be sent to you all.
- Final oral exam to be determined

Course Format

- Several computer exercises, 4 compulsory projects. Electronic reports only using Git for repository and all your material.
- Evaluation and grading: Each project counts 20% of the final mark. The final oral exam counts 20%. A tentative feedback form on projects can be found at the github address of the course
- The computer lab consists of 16 Linux PCs, but many prefer own laptops. C/C++ is the default programming language, but Fortran2008 and Python are also used.
- All source codes discussed during the lectures and for each chapter can be found at the github address of the course. We recommend either C/C++, Fortran2008 or Python as languages.

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
- Ordinary differential equations
- Partial differential equations
- Parallelization of codes
- High-performance computing aspects

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 if time
- Qubic Spline interpolation

Syllabus

Monte Carlo methods (chapter 11).

- Brute force Monte Carlo integration
- Random numbers (simplest algo, ran0) and probability distribution functions, expectation values
- Improved Monte Carlo integration and importance sampling if time.

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 (chapter 8).

- 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.

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.

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

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/analytical form)
* To enable you to develop a critical view on the mathematical model and the physics.

Computing knowledge

Our ideal about knowledge on computational science
Hopefully this is not what you will feel towards the end of the semester!

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