Computational Physics

Projects

David Buscher

Version 3.0 February 2024

Contents

Introduction

The Computational Physics project — if you choose to submit one — is worth one unit of further work, so roughly 10% of your final mark for the year. It involves choosing a problem from the project list below. You will analyse the problem, write and test Python code to investigate it, then write up your work in a report. Like E1 and E2, you can expect it to involve 40 to 50 hours' work. This includes reading and research, coding, experimentation and gathering results, and writing your report.

You may start your project work once this document is published. The deadline for submission of the project report is **16:00 on the first Monday of Full Easter term (29th April 2024)**. Submissions made after that time will not be marked [1]. I suggest you proceed *as if the deadline is 16.00 on Friday 26th April*, and use the final weekend if you really need to.

Bear in mind that **everything in your report should be your own work**, and your submission will be treated as a declaration of this fact. The rules regarding cheating and plagiarism in the Physics course

[1]See the NST rules on late submission. For the avoidance of doubt, the project is a piece of work for which the submission date is specified in Examination Regulations (Ordinances), and therefore the final paragraph applies.

handbook (Page 32) apply here. It is OK for you to use code that others wrote — that's what a library is, after all — but in all cases the attribution should be clear.

The report

Your report should not exceed 3000 words (excluding the abstract, captions, references and appendices) with an abstract of not more than 250 words. **Include a word count at the end of your abstract.**

Please see Keeping Laboratory Notes and Writing Formal Reports for guidance on structure and style.

Include an appendix with an explanation of the structure of the source code that you used for the project and upload as part of your submission (see below). This should include a list of the file names of all the uploaded files and the role they play in the code structure (e.g. main program, plotting program, Python module responsible for main numerical algorithm, etc).

Submission

You should submit your report and ancilliary information on the TiS (https://www-teach.phy.cam.ac.uk/students/courses/computing-project/117). You should upload:

1. A copy of your report as a PDF file. Please give the file a sensible name, something like `project-CRSID.pdf` would be a good choice.
2. A ZIP file (see https://en.wikipedia.org/wiki/ZIP_(file_format)) containing the source code you wrote to do your project, in one or multiple files. See above about documenting your code structure in an appendix to your report. If you include a Jupyter notebook `.ipynb` file as part of your source code, please also include the corresponding plain Python `.py` file (you can use, for example, `nbconvert` to convert between these formats; alternatively, Google Colab also has functions which allow you to download your code in either format).
3. (Optionally) a ZIP file containing any animation files you wish to refer to in your report. If you are uploading animations, be sure to clearly indicate in your report which specific animation is relevant to which particular result — do not expect the assessors to look at any animation unless it is specifically referenced as evidence for some particular result (this is just like the convention for figures - they are only deemed relevant if explicitly referred to in the text). Animations can be in any standard video file format, e.g. mp4. Animated GIF files are acceptable for short animations, but they get large very quickly and there is an upload limit of 100MB.;

Marking

The credit for the project is one unit of further work (like TP1, TP2, E1, E2, etc.). The projects are marked out of ten in the four categories:

1. Analysis of the computational physics aspects of the problem (possible algorithms, their complexity, etc.).
2. Details of implementation of the algorithm and its performance (e.g. in terms of run time on what hardware). This will include the description in the report as well as the accompanying code, and will include a judgement of style, readability (including comments and docstrings) and quality.
3. Results, analysis of errors (if applicable), tests and discussion of the relevant computational physics.
4. Overall presentation of the report, structure, etc.

Although presentation only enters in the last category, bear in mind that assessors will not be able to get a good sense of what you have done unless your work is presented clearly throughout. In particular, it will make your assessor's life easier (always a good thing) if there are **sections that clearly refer to the first three points**.

As a general guide, you should present your work as if your assessor is **completely ignorant** of both the problem and computational approaches to it. Alternatively, present your work as if to yourself before you began working on it.

The projects

For all of these projects the basic protocol is the same:

1. Understand the physical problem
2. Learn about the available algorithms, including their theoretical complexity / performance
3. Implement the algorithm(s)
4. Gather results about the performance of the method

You will need do some research and reading before starting, so follow the links and references[2]. *But read pragmatically*: look for the algorithm description or pseudo-code and start thinking about how you will translate it into your own code.

[2]Most should be freely available (to you) online: don't forget the "CU" in CUP stands for Cambridge University.

Project A: Percolation

Percolation is an extremely simple statistical mechanical model that has been used as a testbed for many theoretical concepts and numerical techniques over the decades.

One of the most popular and efficient algorithms for percolation is Newman and Ziff (2000). Implement and study this algorithm. You may wish to compare the Newman–Ziff algorithm with other algorithms e.g. Mertens (2022).

Project B: Hierarchical methods for N-body simulation

The fast multipole method and the Barnes Hut method are two methods to study pairwise interactions in N-body systems that lower the complexity to $O(N \log N)$ from the naive $O(N^2)$. Implement these methods and compare their performance on different tasks.

Project C: Cluster algorithms for Monte Carlo

Study the performance of cluster algorithms for statistical mechanics models such as the Ising model and hard disks — as described in these lectures — and compare them with the Metropolis algorithm. You should analyze the autocorrelation time of the algorithms and the issue of critical slowing down.

Project D: Lyapunov exponents

Lyapunov exponents quantify chaos in a dynamical system by the exponential divergence of neighbouring trajectories. Sandri (1996) as well as Chapter 3 of Pikovsky and Politi (2016) describe numerical methods for their computation.

It's probably easier to start with *discrete time* dynamical systems, moving on to the continuous time case if you have time. Wikipedia has an extensive list of chaotic maps e.g. the Hénon map. See the section "LCEs of of Discrete Systems" of Sandri (1996).

Project E: Multi-spin encoding for Ising models

In an Ising model the lattice variables can take only two values. The idea behind multispin encoding is to use 64 bit data (`uint64` say) to simulate multiple copies of the Ising model, so that a degree of parallelism comes "for free". This requires *bitwise* operations to be performed on the integers.

Implement and test multispin encoding for the square lattice Ising model. Multispin encoding is introduced in Jacobs and Rebbi (1981) and also described in Section 10.1 of Böttcher and Herrmann (2021).

Project F: The Fermi–Pasta–Ulam–Tsingou problem

A fundamental problem in statistical physics is: how does a mechanical system thermalize? FPUT studied this question in a model of a chain of masses coupled by weakly anharmonic springs. Dauxois and Ruffo (2008) is a nice review of the problem, which also includes sample MATLAB code that you could use as a starting point. You may have to explore other integration schemes (e.g. symplectic integrators).

Project G: Lenia

Lenia is a recently invented (discovered?) cellular automaton: somewhat similar to Conway's Game of Life but with continuous variables.

There are lots of resources on Lenia, including the original paper (Chan (2018)), a GitHub repo, webpage, as well as this article and colab notebook from Google research. Produce your own implementation and investigate its properties!

Project H: Time evolving block decimation (TEBD)

Using Glen Evenbly's site as a starting point, investigate the use of TEBD to solve quantum spin chains. Investigate real and imaginary time dynamics (the latter for finding the ground state), including the evolution of the expectation values of spins, starting from different initial states. See the original paper (Vidal (2004)) for ideas of what to calculate.

Project I: Schrödinger's smoke

Schrödinger's smoke is a recent alogrithm for the simulation of incompressible fluids. The original paper (Chern et al. (2016)) describes the algorithm in Section 2: it is based on the split-step method and FFT as discussed in the FFT lecture.

The project webpage includes Matlab code. *This is an ambitious project*. It's probably best starting with a two dimensional fluid as a proof of principle.

References

Böttcher, Lucas, and Hans J Herrmann. 2021. *Computational Statistical Physics*. Cambridge University Press.

Chan, Bert Wang-Chak. 2018. "Lenia-Biology of Artificial Life." *arXiv Preprint arXiv:1812.05433*.

Chern, Albert, Felix Knöppel, Ulrich Pinkall, Peter Schröder, and Steffen Weißmann. 2016. "Schrödinger's Smoke." *ACM Transactions on Graphics (TOG)* 35 (4): 1–13.

Dauxois, T., and S. Ruffo. 2008. "Fermi-Pasta-Ulam Nonlinear Lattice Oscillations." *Scholarpedia* 3 (8): 5538. https://doi.org/10.4249/scholarpedia.5538.

Jacobs, Laurence, and Claudio Rebbi. 1981. "Multi-Spin Coding: A Very Efficient Technique for Monte Carlo Simulations of Spin Systems." *Journal of Computational Physics* 41 (1): 203–10.

Mertens, Stephan. 2022. "Exact Site-Percolation Probability on the Square Lattice." *Journal of Physics A: Mathematical and Theoretical* 55 (33): 334002.

Newman, MEJ, and Robert M Ziff. 2000. "Efficient Monte Carlo Algorithm and High-Precision Results for Percolation." *Physical Review Letters* 85 (19): 4104.

Pikovsky, Arkady, and Antonio Politi. 2016. *Lyapunov Exponents: A Tool to Explore Complex Dynamics*. Cambridge University Press.

Sandri, Marco. 1996. "Numerical Calculation of Lyapunov Exponents." *Mathematica Journal* 6 (3): 78–84.

Vidal, Guifré. 2004. "Efficient Simulation of One-Dimensional Quantum Many-Body Systems." *Physical Review Letters* 93 (4): 040502.