CS319: Scientific Computing (with C++)

PROJECTS (Version 11.03.25)

Niall Madden

Week 8, Semester 2, 24/25

https://www.niallmadden.ie/2425-CS319/2425-CS319-Projects.pdf

0. Overview

Your project will consist of

1. **Topic:** This is an idea you select/propose, in **one-to-one discussion** with Niall in labs or lectures.
2. **Proposal:** An outline for what you plan to achieve. This will be presented as 250 word project proposal. (Word count is indicative: I won't check).
3. **C++ Code:** Your own implementation of some algorithm, coded in C++ (or C++ and Python), and with your own choice of test problem(s) so you can verify properties of the algorithm, such as its complexity, or convergence, etc.
4. **Report:** ideally written as Jupyter notebook, which explains the problem, describes the algorithm, and draws conclusions about its properties.
5. **Presentation:** lightning talk on your work, including a demo of your code.

Teams of 1 or 2

- You will be a member of a team of 1 (i.e., you) or 2 (i.e., you and one other person).
- For paired projects, you have to find a partner. Then both of you have to let me know.
- You are encouraged to work in pairs if at all possible.
- Both members will get the same grade or all parts jointly done.

Your projects will contribute 40% to your over-all grade for CS319. The break-down of marks is as follows.

Note: all deadlines are at 17:00 on the day mentioned.

What	When	Marks
Project topic negotiated	Noon, Wednesday, 12 March (W9)	5
Project proposal	Noon, Wednesday, 19 March (W10)	5
Presentation	Week 12 (day TBC)	5
C++ Code	5pm, Thursday, 10 April	15
Project report	5pm, Thursday, 10 April	10

1. Project topics

Next, I'll outline a selection of topics you might consider working on. However, I also encourage you to propose your own ideas.

You are required to discuss your ideas with me in the lab tomorrow (6 March) or Friday (7 March)!

Failing that, can discuss by email (though not for full marks, and you will still have to discuss the proposal in person).

Deadline: submit your agreed project topic by **Noon, Wednesday 12 March**.

Some ideas for projects

The topics listed here have the purpose of stimulating discussion. More will be added to this as I think of them. When choosing a topic, you are welcome to suggest one of your own, or pick one of these. However, you must discuss with me. No two people will be allocated the same topic, unless working as a group. Topics can involve standard algorithms in a new setting (e.g., applications of methods studied in class), or new algorithms (maybe to standard examples). Don't assume a topics below is available. Also: the "..." ones are place-holders for ones suggested in class.

1. **Programming:** Mixed precision computing.
2. **Programming:** Parallel or distributed computing.
3. **Programming:** ~~A DBMS~~ [MF]

4. **OOP**: Your own implementation of arbitrary precision integers, with applications, e.g., in cryptography or primality testing.

5. **OOP**: Computing with piecewise polynomials.

6. **Programming/OOP:** ...

7. **Optimization**: ~~Gradient descent in two and more dimensions.~~ [AD+LW]

8. **Data Analysis:** Analysis of tides in Galway bay (or elsewhere): predictions.

9. **Data Analysis:** Analysis of tides in Galway bay (or elsewhere): *how bad was storm Eowyn*?

10. **Data Analysis:** Nonnegative matrix factorization in data science.

11. **Data Analysis:** SVD (=PCA) in data science.

12. **Data Analysis:** ...

13. **Interpolation:** ~~High-order polynomial interpolation with *adaptive* error control.~~ [CS]

14. **Interpolation:** Piecewise polynomial interpolation with *adaptive* error control.

15. **Interpolation:** Interpolation and sampling of functions in 2D.

16. **Interpolation:** ...

17. **Interpolation:** Convexity Preserving Methods.

18. **Quadrature:** ~~Derivation and comparison of 1D schemes, such as Gauss-Lobotto and Gauß-Legendre methods.~~ [CD+ND]

19. **Quadrature:** High-dimensional numerical integration by Gaussian methods.

20. **Quadrature:** High-dimensional numerical integration by hyperbolic cross-points.

21. **Quadrature:** ...

22. **Networks:** Testing planarity.

23. **Networks**: ~~Visualising graphs (e.g., using the Graph Lapacian).~~ [SM]

24. **Networks**: Random Graph Models.

25. **Networks**: ...

26. **Linear algebra**: ~~Schur compliments for solving linear systems.~~ [RC]

27. **Linear algebra**: Estimation of eigenvalues and eigenvectors, e.g., combining The Power Method and the Rayleigh Quotient.

28. **Linear algebra**: QR Factorization.

29. **Linear algebra**: ~~Schur method for eigenvalues.~~ [TD+HT]

30. **Linear algebra**: ...

31. ~~**Differential equations:** Solution of high-order ODEs by implicit Runge-Kutta methods.~~ [SB+CF]

32. **Differential equations:** ~~Solution of boundary value problems by finite difference methods.~~ [EOB]

33. **Differential equations:** ...

34. **Other?:** ...

More sources:

- **Algorithms from the Book**: https://epubs-siam-org.nuigalway.idm.oclc.org/doi/book/10.1137/1.9781611976175
- Projects in O'Leary's book (See Weeks 1 and 2).
- Nick Higham's blog: https://nhigham.com/blog/

2. Project proposal

- The project proposal outlines what you plan to achieve.
- It should begin with a (tentative) title, your name, ID and email address, and the name of your collaborator (if any).
- The proposal should describe what you plan to do, explain why it is interesting, and describe how you plan to test it.
- It should include a set of milestones, and a time-line for each. (It more important that you make such as plan, than actually following the time-line!)
- 250 words should suffice. (Word count is indicative: I won't check).
- This will be uploaded to Canvas as a PDF.
- **Deadline: Noon, Wednesday, 19 March**.
- If your proposal is OK (and on time), you get full marks for that part. If not, you can resubmit until you do. (That is: we want to get this right).

3. The code

The code for your project is a chance for your to show your skills in C++ programming, and in scientific computing.

Examples of things you might try to demonstrate:

- skill in writing modular code. At the very least there should be one function, but I expect there will be multiple functions.
- is your code *efficient*? Have you identified any bottlenecks?
- examples of writing your own class will be very welcome.
- can you use an external data source or produce one (and show skill in data input/output).
- **Reusing unattributed or AI-generated code will be dealt with sternly.**

1. Code should be submitted on Canvas (see `Project` module). If you prefer to share it via a git repository, that would be very welcome. Make sure the repo is private and shared with me. In addition, add a link to it as text in the code section on Canvas.
2. If your code features multiple files, you may submit them as a `zip` file, or similar archive.
3. Only C++ (`.cpp`), Python (`.py`) or Jupyter notebooks (`.ipynb`) files should be submitted. Where appropriate, you can use multiple files (e.g., separate `.cpp` and `.h` files).
4. Every **code** file you submit (other than 3rd party libraries) should, without exception, have comments at the start with
   - your name, ID and email address
   - comments that briefly summarising what the code does and how to use it.

   You are not required to add you name, etc, to data files that are part of the project.

5. There is no lower or upper limit in the number of lines your code must have. My experience is that good projects often feature about 200 lines of code for algorithms, functions, classes, etc, and about 100 lines of code for demonstrating that it works.
6. Code that you claim is written by you should be written by you. Don't use anyone else's code without attribution. *If plagiarism is identified in a project, it will receive a score of zero*.
7. Don't submit AI-generated code. Just don't. You'll probably fail the module with no opertunity to repeat until next year.

Broadly, your code will be assessed for the extent for which it shows your programming in skills in C++ (as well as Python, if appropriate).

In doing that, I will pay particular attention to the following:

- Is there competent use of basic programming structures, such as `for` loops and `if` statements?
- Is there appropriate use of data-types, with conversion between them as needed?
- Have you shown you can work with external data sources, e.g., reading data from text files, or importing from images or spread-sheets, etc.
- **Important:** demonstrated skills in scientific computing, including the implementation of algorithms, and correct use of tools.
- **Important:** You project should have at least one function, ideally several.

4. The report

Your report will be submitted as a PDF document, ideally (but not necessarily) generated from a Jupyter notebook.

It should start with:

- project title
- your name, ID number, email address
- names of anyone you collaborated with.

Suggested organisation:

1. **Introduction:** A summary of what you have done. This should be 300-500 words, and written in a non-technical style: anyone should be able to understand it. The emphasis should be on the problem solved, and why it is interesting, and not on the code.

4. The report

2. **Code:** A technical discussion of the code. Outline the major algorithms you've implemented, functions you've written, or classes you've developed. This can be quite "high-level": you don't have to explain what individual lines of code do, but rather focus on the big picture: *what was the main challenge to be addressed?*
   However: if you code makes use of any aspect of the C++ language that was not covered in class, you should explain that. If you may present this part as a video with a walk-through of the code.
3. **Case use:** An example of typical input and output. Having read this, the use should know enough to test your code for their own examples.
4. **Discussion:** A discussion on what you have learned/discovered. What have you discovered about this algorithm and/or this method? **What steps have you taken to verify its properties?**. For example, if relevant, have you established the rate of convergence, or estimated the run-time for inputs of given sizes?

4. The report

5. **Highlights:** What are the highlights of your project? What took the most effort? What are you most proud of?
6. **Conclusion:** Details any limitations of your code that you have noted, and a comparison with your original project proposal. If you had more time to work on your project, what would you do next? List any references used.

For group projects:

- Each member of the group must submit copies of the code and report.
- The report must state the members of the group, and **what aspects of the project that each contributed to.**

4. The report

Submit a PDF version of the report (which will be be checked with TurnItIn). You can also upload a Jupyter notebook, if you think it would be useful.

Important: you are also welcome to upload a short video explaining how your code works. This might be particularly useful if your code contains any constructs that were not covered in class.

- Presentations will take place in Week 12.
- Each presentation should feature at most 3 slides. The topics might be
  - My project topic
  - What I've learned so far
  - What I plan to do next
- Joint projects will involve joint presentations. Both partners must present (e.g., alternate slides).
- If you like, you can demo your code, providing you plan are confident you can connect to the projector.
- Presentations should take at most 5 minutes. Each will be followed by a question.

come back later