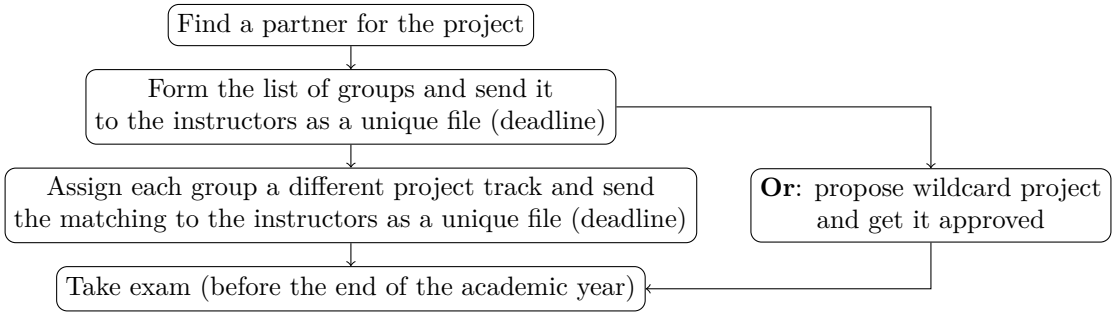
Information on the final exam

Computational Mathematics for Learning and Data Analysis (646AA), Università di Pisa

The exam for this course is composed of two parts: a take-home project, typically performed in pairs, and an oral exam.
Projects come in two categories: ML (on topics that overlap with the Machine Learning course) and non-ML (on topics that concern this course only). You may choose the one you prefer.

1 Group and project selection

We will give deadlines to form groups and subdivide projects and announce them on Moodle and Teams; the process will take place around November. The process works as described in this diagram.



Latecomers who did not participate in the initial pairing stage and/or groups → projects matching will still be able to form groups and/or get a project assigned at any later point in time, but they will clearly have to choose (on a first-come, first-served basis) among the tracks that have not been taken already.

1.1 Pairing up

We expect the students to form groups of 2 persons each. Singleton groups may be allowed with convincing motivations (e.g., being away for Erasmus, no available partners, . . .) after discussion with us. Three-students groups can also be allowed, upon request, with *significantly heavier* wildcard projects.

After the deadline for forming the groups, we expect in return a single file with a list of *all* groups in a format such as

```
Group  e-mail                            matricola
1      giovanna.bianchi@studenti.unipi.it 601235
1      farhad.davani@studenti.unipi.it    601234
2      paolo.russo@studenti.unipi.it      595147    # singleton
3      lin.xiang@studenti.unipi.it        612532
3      marco.meucci@studenti.unipi.it     598214
...
```

1.2 Subdividing projects

After the groups are formed, we will provide a list of suggested project tracks. The students will have time (at least one week) to agree collectively on how to subdivide the projects so that *every group performs a different project*. We expect to receive a single file with the groups → projects matching with a format such as:

```
Project track number    Pair number
ML-1                    2
ML-2
ML-3
...
ML-30                   1
No-ML-1                 3
No-ML-2
...
No-ML-25                4
```

We expect an *unanimous* agreement to be reached on the matching; otherwise, if even a single student reports dissent from the decision, we will step in and give a randomized assignment (which would most likely leave more people unsatisfied). We will only guarantee to keep your preference between ML and non-ML projects (which means, two partial randomized assignments).

In case there is a shortage of projects in one of the lists, we will add more.

1.3 ML and non-ML projects

Project descriptions consist of

- one or more *problems* to solve;
- one or more *algorithms* to be used to solve them.

There will be two lists of projects:

- *ML projects*, on problems that overlap with the topics treated in the Machine Learning course: neural networks and support vector machines;
- *non-ML projects*, which are self-contained on the topics of this course.

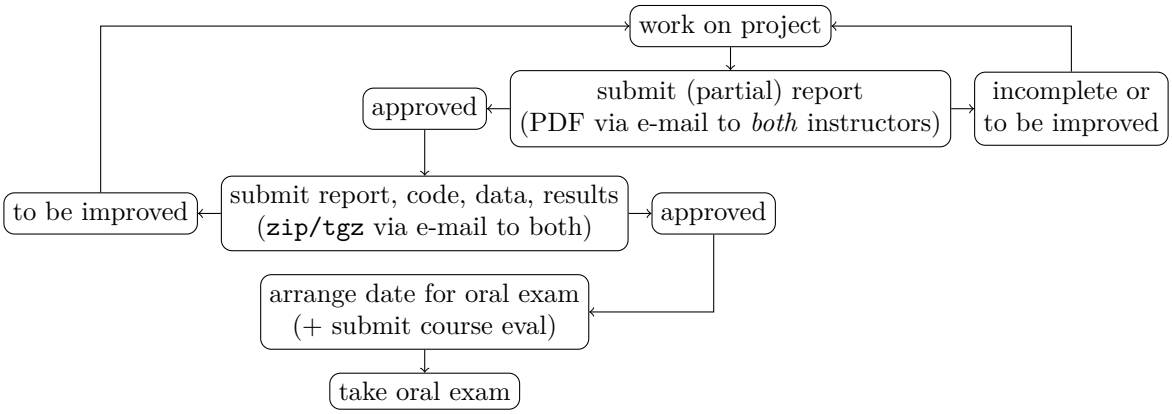
You may choose among either list. ML projects benefit from some ML background, but apart from that there is no difference in the difficulty of the two groups of assignments, nor in the grading.

You may be able to re-use parts of the code for this project for your ML exam (further instructions by prof. Micheli). Note, though, that the two exams (and the two reports) are quite different: in this course we focus on the mathematical side of algorithms (quality of the obtained solutions, in terms of "gap" or "residual", and efficiency in obtaining it), while ML typically focuses on the usefulness of the algorithm in learning, which is a completely different concept. So, while performing a ML project can have a few advantages if you follow both courses, it is not substantially different from a non-ML one.

There will be also be the possibility to propose "wildcard" projects where you either mix-and match between problems and algorithms proposed in the list in a way that is not done in the existing projects, or propose your own problems / algorithms. This is very welcome, and an accepted way to justify the proposal of forming three-students groups, provided the content of the project warrants it (that is, significantly more problems and/or algorithms than in standard projects, or significantly more complex ones). However, wildcard projects are not intended, and must not be used, to clone existing projects from the list, even with minor modifications.

2 Workflow of the exam

The process works as described in this diagram.



Please communicate only by e-mail. E-mails should *always be sent to both lecturers*, even if the content seems to be "exclusively numerical analysis" or "exclusively optimization".

Submitting the report (and having it accepted) is typically the most time-consuming part of the process. We *strongly suggest* that you send us your reports in partial instalments. Each report should contain a number of separate parts (see the next section for details), and a good practice is to send us the report each time you complete one of them. In this way we can check that you are on the right track, correct conceptual mistakes or suggest improvements. Some "feedback loops" are normal; usually we *will* ask for some changes. Getting feedback early may allow you to avoid building on erroneous assumptions and having to redo large parts of the project. In particular, we strongly suggest to complete the "theoretical" part of the report before you invest any serious amount of time into implementations. However, if you really want to take the chance to let us have the report all in one blow when you think it is complete, be our guests; only, do not give it for granted that we will immediately approve it, and that you will immediately get access to the oral exam.

Project reports (complete or partial) can be submitted at any time until the start of the next version of this course in the next Academic Year (AY). At that time, projects that have already started but are stalled for some reason must be completed before the projects of the new AY are assigned. Upon request, we may grant a further extension to groups that have already performed a significant amount of work, as demonstrated by their partial reports, with the idea that they finish "soon" (say, before the end of the calendar year).

Once the complete report is approved, and *only at that moment*, you are supposed to let us have the final distribution of your code and of the supporting material in a single `.zip` or `.tgz` file and have us accept it: auxiliary files, batches or scripts required to run it, README, data files, logs / csv files / spreadsheets with the detailed results of the experiments, ... Typically this will be approved right away, since all serious issues should hopefully have been ironed out in the previous phase, but we still reserve the right to check and examine it and ask for improvements, say, in comments or in the accompanying material, or even more serious work in the case where the code reveals issues that the report had not. Once both the report and the code are approved, the oral exam can be held and the grading finalised.

Please remember to fill in the course evaluation questionnaire *before* the oral exam. It can no longer be done once the exam is finalised, and we cannot check it because we don't ask you to register for the exam (as most other courses do). Your feedback is important to the continuous improvement of the course: this is a service to your fellow students of the subsequent years, so do not forget it for their sake.

The project will contribute substantially to your final mark. Once the project is assigned it cannot be changed, except for extraordinary reasons to be discussed with us, nor is it possible to require a second project to improve your marks up until the end of the process (the re-starting of it in the next AY).

3 Avoid plagiarism

While performing your project, you may look for relevant theoretical results in papers, websites and books, but *all text must be written by you.* Copying from existing material (slides, books, papers, internet) is *not* allowed and will be mercilessly crushed upon. The term is intended in a wide sense: not only verbatim copy of text, formulas and images, but also thoroughly borrowing someone else's material and structure, even after editing to make the copy superficially different from the original. You must start from a blank sheet of paper (or file) and explain things with your own words. The exception is that you may (and should, whenever appropriate) repeat definitions and theorem statements word-by-word.

In any case, your sources should be explicitly referenced: "This theorem is from [1, Theorem XYZ]", "We are following the approach in [2, p. XYZ]".

In particular, carefully avoid to "take inspiration" from projects reports by past students; we have them all and we check. Similarly, when you are finished please do not share your project with future students of this course (e.g., if you are using a `git` repository do not make it public): the challenges in this project are meant to be solved without relying on previous solutions. You would do a serious disservice to them on two different counts: preventing them from acquiring the notions and the knowledge you have developed over the course of the project, and especially putting them at serious risk of formal disciplinary action. We take plagiarism, copying and academic misconduct seriously, and we will defer any student found guilty of such misdemeanour to the university's ethics panel. Ask the instructors if you have doubts on originality and plagiarism issues.

4 Structure of your report

4.1 Setting the stage

The first section of your report should contain a description of the problem and the algorithms that you plan to use. This is just a brief recall, to introduce notation and specify which variants of the algorithms you plan to use. Your target audience is someone who is already familiar with the content of the course. There is no need to repeat a large part of the theory: we are sure that you know how to do that, given enough time, books, slides, and internet bandwidth.

If adapting the algorithm(s) to your problem(s) requires some math (such as developing an exact line search for your function, when possible, or adapting an algorithm to deal more efficiently with the special structure of your problem), you are supposed to discuss it here with all the necessary mathematical details.

Discuss the reasons behind the choices you make (the ones you can make, that is, since several of them will be dictated by the statement of the project and should not be questioned unless you think you have found a serious conceptual flaw in our decision).

4.2 What to expect from the algorithm(s)

Next, we expect a brief recall of the properties that you expect to see in the experiments; for instance:

- Are there any results that may guarantee the convergence of your algorithm? At which speed? Monotonically? Are the hypotheses of these results (convexity, compactness, differentiability, etc.) satisfied by your problem? If not, what are the "closest" possible results you have available, and why exactly are they *not* applicable? Do you expect this to be relevant in practice?
- If the algorithm is not iterative, is it going to be stable and return a good approximation of the solution?
- What is the complexity / efficiency of the algorithm?

Each time you use some specific result (say, a convergence theorem), please be sure to report in detail what the assumptions of the result are, what consequences exactly you can derive from them, and the source where you have taken it (down to the number of theorem/page). Discuss in detail why the assumptions are satisfied in your case, or which assumptions are not satisfied or you cannot prove they are.

4.3 Write your code

Coding the algorithms is a major part of the project. Languages that are often convenient for numerical computation are (depending on the task) Matlab, Python, Julia and C/C++, but you can use any reasonable programming language.

You are expected to implement the algorithm yourself; it should *not* be a single line of library call. However, you can use the numerical libraries of your language of choice for some of the individual steps: for instance, you can use `numpy.linalg.norm` to evaluate the error, or Matlab's `A \ b` to solve a linear system that appears as a sub-step (unless, of course, writing a linear solver to compute that solution is a main task in your project).

You can (and should) also use numerical libraries to compare their results to yours: for instance, you can check if your algorithm is faster or slower than Matlab's `quadprog`, if it produces (up to a tolerance) the same objective value, or how the residual of your solution compares with that produced by `A \ b`.

When in doubt if you should use a library, feel free to ask us.

Your goal for this project is implementing and testing numerical algorithms: software engineering practices such as a full test suite, or pages of production-quality documentation, are *not* required. That said, we appreciate well-written and well-documented code (who doesn't?). You are free to use tools such as `git` to ease your work, if you are familiar with them, but giving us a pointer to the `git` repository is not the expected way to communicate with us (especially since you should never make it public).

4.4 Choose and describe the experimental set-up

Next, we expect a brief description of the data you will test your algorithms on. For "ML projects" this will typically be provided by the ML course, but still a modicum of description is required. For "no-ML projects", it will typically have to be either generated randomly, or picked up from the Internet, or a combination of both. This is not always trivial: the random generation process can be tweaked to obtain "interesting" properties of the data (what kind of solution can be expected, how well or ill-conditioned the problem is, ...). These aspects should be described in the report.

You are supposed to test the algorithm on a realistic range of examples, in terms of size, structure and/or sparsity: it is typically *not* OK if your largest example is 10×10. Get a sense of how algorithms scale, and what is the maximum size of problems that you can solve reasonably quickly on an ordinary machine.

Numerical experiments have two purposes:

- Confirm that the algorithms work as expected: how close do they get to the true solution of the problem? How can you check it? Is there a "residual", or "gap" value that you can check? Do they converge with the rate (linearly, sublinearly, superlinearly, ...) that the theory predicts? Does the error decrease monotonically, if it is expected to do so?
- Evaluate trade-offs and compare various algorithms: which algorithm is faster? If algorithm A takes fewer iterations than algorithm B, but its iterations are more expensive, which one is the winner? How does this depend on the characteristics of the problem to be solved (size, density, ...)?

When designing your experiments, and later your graphs and/or tables, you should have these goals in mind. To quote a famous mathematician, "the purpose of computing is insight, not numbers."

Comparison with off-the-shelf software is also welcome (and often useful to check correctness) to assess whether your approach could ever be competitive under the right conditions (and what these are).

Setting thresholds and algorithmic parameters is a key and nontrivial aspect. This is one of the "dark secrets" of numerical algorithms: basically any algorithm you can write has parameters that can have a huge impact on performance, and it will misbehave if you do not set them properly. Thus, a minimal testing activity about the effect of these parameters is almost surely needed. A full-scale test à-la hypermetameters optimization in ML is also possible, and welcome, but this is anyway different from the one in ML, even for "ML projects". The properties of interest are fundamentally different: in ML one looks for learning (accuracy, recall, ...), while in this course one is looking at

"how close the solution I got is to what I would have liked to get, and how costly was it to get there". Hence, reporting here the same hypermetameters optimization tables done for ML does not cut it. Not that these *cannot* be reported, as they still give some potentially interesting information, but it is not the information we are interested in.

4.5 Report and comment the results

A few plots and/or tables are required to display the results of your experiments. Have a purpose in mind: as for the choice of experiments, each plot should display one or more features of the algorithms: convergence speed comparison, residual, CPU time scaling, ...

Plots should be readable. If 90% of your plot are barely-visible horizontal or vertical lines, look for a better way to display information. *Logarithmic scales* are usually a good idea to display quantities (such as residuals or errors) that vary by orders of magnitude. In particular, they display well convergence speed, since a sequence with exact linear convergence ($\text{error}_{k+1} = C \cdot \text{error}_k$) becomes a straight line.

For "ML projects", the quantities you want to plot may be quite different from these you would in a ML course. While the "learning rate" (here "convergence rate") is the same, the out-of-sample performance is much less relevant for us. Always keep in mind what is the information that is important for you to show and do your best to show it effectively and efficiently: three pages-long tables of 7pt figures or 10 pages filled of very many small plots are typically neither efficient nor effective at conveying the information to your readers.

5 The oral exam and grading registration

Once your project report and the final distribution of the code are accepted, we will agree with you a date and time for the exam. There are no "appelli" for this course: you may safely ignore the dates on `https://esami.unipi.it`, and there is no need to register for the oral exam when we agree upon the date. However, *please do submit your course evaluation* on `https://esami.unipi.it` before the oral exam (or earlier). Since we don't require you to register we have no way to force you to do it, but it is very important feedback for us. Letting us know what works and what doesn't in the course is helpful for your fellow students of the following AYs.

In general we will be available to check your (partial) reports and make the oral exam at any time of the year, but keep in mind that we have other commitments, too (say, in August our reaction times could be significantly slower). Typically all students of a group are expected to take the oral exam at the same time, but exceptions are possible upon well-motivated request.

The oral exam will start with a discussion of your project, and possible improvements. Usually, other parts of the course's syllabus emerge during this discussion. You are expected to be familiar with all the main ideas and topics of the syllabus, but the main purpose of this course is insight, not proofs, and this is what you will be tested on.

If you agree with the final grading we propose you, it will be immediately registered in the system. It is of course always possible to refuse the vote that you are offered. However, an accepted project cannot be changed or improved. Hence, the only way to improve your vote is to re-take the oral exam. If this happens, the exam can obviously no longer focus on the project, and therefore it will necessarily have to go more in detail on the mathematical theory of the course. Hence, should you want to try to improve your grading this way, you are better make sure that you return well prepared on the theoretical part. There is no guarantee that a new oral exam will improve the vote that you are finally offered.

6 Final remarks

Interacting with the lecturers during the development of your project is a useful mean of discussing and clarifying the contents of the course, especially of those more strictly related with the project itself; we suggest you to exploit this option. However, this is not a good substitute of asking questions during lectures and office hours (ricevimento): we suggest you to exploit these options, too. We are happy to discuss the course topics at any time (though we have other teaching, research, and organisational tasks). Please always address *all* e-mail to both of us: the course is one and the same.

Good luck! We hope that you will enjoy the course, and that it will be useful in your future career.

Federico and Antonio