

About your project

Goal. The goal of the Math 661 project is for you to become more familiar with certain optimization problems and algorithms than is possible with the brief coverage that is typical of the rest of the course.

Expectations. Your project may be application-driven (*choose the problem(s) first*) or algorithm-driven (*choose the algorithm(s) first*)—see page 3—but all projects must include a specific optimization problem and a specific optimization algorithm. That is, you will implement at least one algorithm, i.e. in MATLAB/PYTHON/ JULIA/ etc., and apply your code to at least one example problem.

Both numerical computation and mathematical analysis are required. For the latter you will analyze the problem(s) and algorithm(s) using theory from the textbook¹ and/or from other references. Once your code is running you should provide some empirical (numerical experimentation) evidence regarding the error and/or performance of your algorithm(s). Analysis is important because it shows you have absorbed ideas from the course, and because it distinguishes between algorithms. Numerical evidence is important because it confirms that you understood the algorithm well enough to implement it correctly.

Your problem(s) must be in the following form:

$$(1) \quad \min_{x \in \mathbb{R}^n} f(x) \quad \text{subject to} \quad \begin{aligned} g_i(x) &= 0, & i \in \mathcal{E}, \\ g_i(x) &\geq 0, & i \in \mathcal{I}, \end{aligned}$$

though of course you may replace min with max. If your project is algorithm-driven then you must identify which such problems are solved by your algorithm(s).

Form (1) describes a very large class of problems, but note your problem(s) must be finite-dimensional (\mathbb{R}^n), though they may arise from an infinite-dimensional source. The problem(s) must be well-enough understood to allow you to both precisely identify the objective function $f(x)$ and precisely identify a feasible set S defined by finitely-many constraints $g_i(x)$. It is o.k. if there are no constraints.

Due dates. There are two due dates for the project:

I = Proposal: **Part I is due Friday 9 November at the start of class.**² There are no format requirements for this part except that it **can be at most two pages**.

The content of the proposal should be to say precisely what problem(s) or algorithm(s) you want to look at. Several quality references are *expected*; online references are o.k.³ Make specific references to our textbook when that is appropriate.

You should spend at most (say) 6 hours on this. However, spending at least a few hours on thinking and research at this stage can be very effective.

¹Griva, Nash, and Sofer, *Linear and Nonlinear Optimization*, 2nd ed., SIAM Press 2009.

²This is a change from the date on the syllabus.

³However, I find that many informal online documents are of low quality.

Ideally you will talk about and play with ideas as you get a clearer idea of what to write in your proposal.

It would be good to use a version control system if you know how. In any case, your efforts should grow in an organized manner and not get lost.

II = Project: **On Tuesday 11 December at 5pm you will submit the complete project.**

It should have the format and section headings as shown on page 4 below. This format corresponds to a L^AT_EX document template already posted online, but of course you do not have to use L^AT_EX:

bueler.github.io/M661F18/blankproject.tex

The total length **must be 20 pages or less**; I will not accept longer projects. The total time spent on the whole project should be at most 25 hours.

Choosing a topic. One of my jobs will be to help you choose problem(s) and algorithm(s) so that your project has reasonable difficulty. I will likely advise you not to bite off too much! It is very easy to get lost in the application, the algorithmic details, or in difficulties with codes or analysis.

You may not choose a topic which has been, or will be, adequately covered in the course. For example, though the basic simplex method is not a good topic, implementations of the simplex method which respect sparsity are (Chapter 7). Similarly, the classical Newton method is not a good subject but investigating quasi-Newton methods, line search methods, or trust region methods may be quite good choices (Chapters 11,12). Your proposal allows me to give good feedback on this, perhaps a gentle nudge in the direction of a nice variation to study.

Here are three approaches to choosing a topic if you don't already have one:

Approach 1: Inspiration from Wikipedia page on mathematical optimization. See the "Applications," "Major subfields," and "Computational ... techniques" sections of

en.wikipedia.org/wiki/Mathematical_optimization

Approach 2: Investigate skipped material from the textbook. Consider choosing a section(s) that you find interesting and which we did not cover. I am not really recommending just choosing a section at random; the Wikipedia page above is a better if you are starting from scratch.

Approach 3: Talk to your advisor and/or do something related to your thesis. You can talk to me, but it will take a while for me to know what you are interested in. It would be better to talk to your thesis advisor (if you have one). It is reasonable to ask "are there optimization problems related to my expected thesis"? Broadly-speaking these might relate to optimal design or parameter fitting, or to algorithms which arise in your field of interest. There may be a paper to read about optimization in your field.

Structure of the project. Here is a rough flow-chart.

