EE364b Projects

Stephen Boyd

January 7, 2007

The EE364b project is meant to be substantial, involving some combination of independent research, implementation and testing, simulation and verification, and documentation. You can propose anything you like as a project; the descriptions below are meant only as broad categories and generic examples. We might compile a list of possible projects a bit later. We're also happy to make some suggestions for projects.

1 Formal project requirements

- *Initial proposal.* Due end of third week. This should include the topic, team members, and some ideas about the approach. Include some background, basic problem, and references. Suggested length: 1–3 pages.
- *Midterm progress report.* Due end of sixth week. At this point you should have a good idea of your approach, and some preliminary results. Suggested length: 5–10 pages.
- *Final report.* Due end of 9th week. Please try to keep the body of the report under 15 pages.
- *Final presentation.* These might be in the form of a poster session.

All formats must be universally readable, such as pdf or simple html. *No proprietary formats (word, powerpoint) will be accepted.* We'll make some latex templates available for you. If you end up typesetting more than a small amount of mathematics, there is really no other choice.

Your report and other supportong materials should be assembled into a simple self-contained web site, with a master index.html file with links to the other documents, codes, and other materials. Our goal is to post these on the web.

2 Types of projects

2.1 A new modeling approach

The most common project will involve an innovation in *modeling* (in the sense used in optimization), *i.e.*, the way a practical problem is formulated as an optimization problem. In such a project, you develop a new approach to some engineering (or other practical) problem, possibly in simplified form. Such a project will likely have the following components:

- *Background and problem in general terms.* You must clearly describe the engineering problem, giving the background, describing how the problem is solved now, and what's lacking, or inadequate, in how it's done now. For example, current design practice might be ad hoc, ignore a number of constraints, etc.

  This description can be vague, as in "The goal is to estimate the original uncorrupted signal, without excessive sensitivity to noise." Don't mix in anything here that is related to how you're going to solve the problem.

- *Your new formulation.* Explain your formulation of the problem (or some part of it, or some simplified variation) as a convex optimization problem (or game, or convex optimization based heuristic), etc.. Be clear about what the variables and constraints are, and whether the problem is convex, or nonconvex. Is your modeling (formulation) accurate? What constraints and specifications can your formulation handle? Which can you not handle? For which is your method heuristic or approximate?

  If the problem you formulate is nonconvex, how will you solve it, or approximately solve it, using convex optimization? If you're not solving the problem exactly, you will need to justify that the method works. This might involve simulation of examples to show the method works well, even if it's not known to be optimal.

- *Verification.* You may need to demonstrate that despite the approximations and simplifications you made in formulating your problem, or in solving your problem (if it is not convex), the end result is still useful (and hopefully good). If you have a new method for design of some circuit, for example, you might provide some SPICE simulations of the designs produced by your method. If your method involves a heuristic for solving a hard (say, combinatorial) problem, then you can consider some cases that are small enough for you to compute the global optimum, which can then be compared to the approximate solutions produced by your method.

  It might be appropriate to compare the results your method to existing ones (and hopefully, show a dramatic improvement).

To solve the problem you formulate you can use standard codes or modeling languages such as CVX, or (if appropriate) develop your own algorithm and solver. If the problem you formulate has structure that can be exploited, and very large problems (or very short solve times) are of interest, you can develop an efficient implementation. If appropriate and interesting, you can develop a distributed algorithm for solving your problem.

We emphasize that your formulation does not have to be a convex optimization problem. If it is nonconvex, however, you must use some convex optimization based methods for solving it, perhaps approximately or heuristically. You could form a relaxation, use a randomized method, use branch and bound (with convex optimization based lower bounds), or repeated linearization/convexification to (approximately) solve your problem. However, it is *unacceptable* to simply form a nonconvex problem and then approximately solve it using some standard nonlinear programming method or code.

Generic titles for projects of this style are "Optimal XXX via convex optimization", or "Optimal design of XXX via convex optimization". Some more specific examples:

- Color correction via convex optimization
- Coordinated vehicle trajectory design via convex relaxations
- Transmit filter design via convex optimization
- DRAM cell design via geometric progamming
- Optimal power assignment in wideband FDMA system with nonlinearities

The principal final artifact for a project of this type is a research paper, along with all supporting material such as (well documented) codes, scripts, and data.

2.2 Pedagogical project

In a *pedagogical* project, the goal is not (directly) to do original research. Instead the focus is on creating materials that are meant to explain, illustrate, or illucidate some aspect of convex optimization, or some application of convex optimization.

For example, you might work out or collect from the literature the details of some application (say, digital filter design), develop simple codes to carry out the designs, and then work out a number of interesting examples. You might develop a simple python application for interactive digital filter design.

As another (more generic) example, you might develop implementations of some algorithms for convex optimization, that are optimized for clarity and simplicity (not performance), that students in future years can use.

Some sample titles:

- "Digital filter design"
- "Trajectory optimization"
- "Antenna array weight design"
- "Experiment design"
- "Extremal volume ellipsoids"

- "Hyperplane classification"

Final artifacts for such a project: A well written lecture, a set of lecture notes, a set of routines that carry out the designs (perhaps a small software toolbox); many examples worked out with nice trade-off plots. Good writing and clear documentation is very important in such a project.

As an extreme case of this type of project, you could commit to making a number of wikipedia entries in some specific areas related to convex optimization.

2.3 Algorithm/computational/software project

You will implement one or more algorithms. You must decide what to implement, and in addition to developing the code, you must develop documentation, test suites, etc. The goal is to make the software available publicly. The software can be in Matlab/octave, C/C++, Python, or even R. It should be as simple as possible, totally portable. You should avoid, if possible, using anything but public domain code.

You can develop low level code (calling LAPACK, BLAS, or some sparse libraries), or build on top of, or contribute to, higher level software such as CVX (Matlab), CVXOPT (Python), or R.

One variation on this type of project is to develop new algorithms for solving large-scale instances of some family of problems, *e.g.*, from machine learning, signal processing, or finance. You will exploit problem structure and use some tricks to solve really big problems, beyond the reach of standard methods and software.

Final artifact: A directory, to be made publicly available, that contains the source code (makefiles, etc.), and possibly executables for several architectures, very clear documentation, and a number of examples and test routines.

Samples:

- "C/LAPACK implementation of Newton and barrier methods for dense problems"
- "Algorithms for large-scale total variation reconstruction"
- "Python package for distributed convex optimization"

2.4 Theory

Of course, this is not the main focus of the course. But we are open to a well-conceived project focussing on some theory related to the course material, *e.g.*, self-concordance and complexity analysis, relaxations, randomized algorithms.