**CS 169/268 Introduction to Optimization**

**Syllabus, Fall 2023**

**Instructor: Eric Mjolsness**

emj@uci.edu

**TA: Stelios Stavroulakis (sstavrou@uci.edu).**

**Help Session ICS room 458A Wednesdays11am-12pm**

**Office hours:TBD.**

**Instructor office hours: Thursdays 3-4:30pm (excluding October 5).**

This course will be in-person only. Real-time in-class participation is required.

## Resources

***Canvas web site!*** (Main organization point for the class including announcements and various integrated tools. Make sure that announcements reach you within 24 hours.)

Other tools: Google UCI G-suite Docs, Sheets, and Jamboard (group work)

**Rough outline of topics (*= advanced: for grad students; optional for undergrads):**

1. Introduction to (introduction to) optimization
    a. 1D unconstrained opt methods
    b. problem definitions
        i. zoo of opt problems
            1. discrete vs. continuous *(& interior point methods)*
            2. local vs. global opt; local vs. global convergence
            3. unconstrained vs. convex vs. constrained
        ii. structure & topology of spaces (domain, range, function spaces)
        iii. special cases in opt:
            1. linearity, quadraticality, convexity
                a. in objectives vs. in constraints
            2. sampled objectives, differentiable models & programs, …
            3. integer- or discrete-valued solutions
                a. combinatorial optimization
2. Unconstrained optimization
    a. optimality conditions
    b. nonderivative methods
        i. Nelder-Mead, Simulated Annealing, Genetic Alg.s, Diff. Evol.*, …

    c. convergence rates & condition number
    d. gradient methods (conjugate gradients; block coordinate descent; …)
        i. Preconditioning *
    e. Newton & quasi-Newton methods
    f. sampled objectives: Levenberg-Marquardt and Stochastic Gradient Descent
    g. multiscale/multigrid methods *
3. Equality-constrained optimization
    a. optimality conditions: Lagrange multipliers
    b. gradient projection
    c. augmented Lagrangian method, & lasso *
4. Discrete optimization
    a. combinatorial optimization
        i. Linear programming: Simplex and Interior Point methods
        ii. branch and bound *
        iii. linear & quadratic assignment
        iv. computational complexity & NP-completeness *
    b. interior-point methods
5. Inequality-constrained optimization
    a. optimality conditions: Kuhn-Tucker
    b. barrier & penalty methods
    c. duality
    d. gradient methods eg. active sets *
    e. convex optimization *
6. Application areas *
    a. graph-defined optimization problems
    b. machine learning (*but* in this class, statistical aspects are not the focus!)
    c. logistics & operations research - e.g. traffic flow
    d. structural mechanics, e.g in mechanical, civil, and biological engineering
    e. computer vision and image analysis
    f. robotic planning, at multiple levels
7. Advanced topics (** probably not in class, but investigate for final project?)
    a. Alternating Direction Method of Multipliers, in parallel
    b. Algebraic multigrid - scaling up
    c. Symbolic regression
    d. Nondifferentiable problems:
        i. subgradient methods
        ii. cutting planes
    e. application family: Finite Element Method
    f. continuation/homotopy methods*
    g. Constraint-specific methods, eg. orthogonal matrices

**268 Variant of the class:**

A major goal for 268 this year will be to get to ADMM (alternating direction method of multipliers) and AMG (Algebraic Multi-Grid) methods. If you know these well already, the course may be too elementary for you. Additional related milestones are preconditioning, optimization by and for machine learning, parallel optimization algorithms, and problem formulation with compound indices. There will be readings from advanced sources in support of these goals. Readings for 268 will continue through the later parts of the course.

**Assignments and Grading:**

For undergraduates : There will be about four homework sets worth 35%, 3 quizzes worth 25% total, a class participation grade worth 10%, and a group project worth 30%.

For graduate students: There will be about four homework sets worth 35%, 3 quizzes worth 25% total, a class participation grade worth 10%, and a group project worth 30%. The graduate student work will be more extensive by about x2, and more advanced.

Programming will be in Julia (with a Python backup), except possibly for the final project.

For each assignment, you will have to submit a pdf report (ideally generated from your notebook file). Additionally for each assignment, upload all materials (code and report) on Canvas. Please follow these guidelines unless otherwise stated.

The Group Final Projects will be described in a separate document.

See "Workload" below.

**Schedule:**

The midterm exam is scheduled as follows:

**Monday November 6**

It will cover theoretical material such as the derivation and properties of inner-loop optimization update equations. It will be closed book and without electronic aids.

Instructor absences: No class day absences currently scheduled.

*Final Projects due at the time of the final exam:* poster presentation on ***Monday, December 11*, 4-6pm** which is at the regularly scheduled Final Exam time for this class; and all electronic materials submitted by 1**1:59pm** the *previous evening* **December 10** (Sunday). So you leave the poster session done with class. *Attendance*

*is mandatory for all students* (not just representatives of each group), and a written group report due together with source code and an electronic form of the poster are due the previous evening. *Do not make scheduling reservations that conflict* with this event or with the evening deadline. If you can't be available then, don't take the class. All group members must be named.

**Reference Works Needed**

Three books, which should all be available in the UCI book store (physically) or for purchase and shipping online. Note that the bookstore required/recommended listing is *incorrect* for 169.

***1. Required*** *(we will use the code from this book - not enough theory though):*

Kochenderfer and Wheeler, "Algorithms for Optimization", MIT Press.
https://mitpress.mit.edu/books/algorithms-optimization

***Slides available*** at the above site, or follow this direct link: https://web.stanford.edu/~mykel/algforopt/slides/ . We will use many of these slides - when in doubt, check here first.

***2. Recommended*** *(more theory):*

A. Belegundu & T. Chandrupatla, Optimization Concepts and Applications in Engineering, 3rd (2019) ed. Cambridge U. Press.

*3.* ***Optional*** *(deep theory - highly recommended for grad students):*

D. Bertsekas, Nonlinear Programming, 3rd ed. (2006) Athena Scientific. (For more serious students of numerical optimization.)

Graduate students should get the required book ***and*** the other two books.

*Alternatives and background reading on special topics:*

S. Boyd, Convex Optimization. (Somewhat specialized to … convex optimization.)

D. Luenberger, Linear and Nonlinear Programming. (A bit dated, but classic.)

W. Press et al., Numerical Recipes, Cambridge U. Press. (Lots of actual, useful working code available online.)

Other readings will be provided in ADMM, AMG, symbolic regression, convergence theory.

**Software resources**

**Julia (**with Python as backup) will be language for all or nearly all homework problems. There are extensive numerical packages available for each. For Julia you will need Optim.jl and/or JuMP for K&W-compatible optimization, and something like Plots for visualization, at least; and optionally later on, SymEngine (as in K&W book) and/or the new Symbolics.jl for computer algebra, Flux for machine learning, and/or tools for automatic differentiation and for differentiable and/or probabilistic programming. For Python you will need numpy, scipy, and matplotlib; and optionally later on, scikitlearn for machine learning and sympy or symengine for computer algebra.

For group final projects, open or academically licensed software such as Mathematica, Matlab, R, … mathematical Problem-Solving Environments (PSEs) have built-in optimization capabilities. Gurobi is specific to optimization and has a free student license. AMPL is a higher-level optimization language.

**Mathematics**

After brief review, and pointing to resources allowing an energetic student to catch up from a lower level of preparation, we will assume the following:

- A working knowledge of calculus (mainly derivatives), linear algebra (e.g. $A \cdot \mathbf{x} = \mathbf{b}$; $A \cdot \mathbf{x} = \lambda \mathbf{x}$ ), multivariable calculus (e.g. multivariable Taylor's theorem; introducing compact notation such as $\partial_j x_i = \delta_{ij}$),
- also some lightly reviewed matrix theory (e.g. SVD = singular value decomposition of a matrix); also discrete mathematics, ability to read and write a proof in logic notation, subscript/index notation (e.g. $[A \cdot \mathbf{x}]_i = \sum_j A_{ij} x_j$ ; $\sum_j \delta_{ij} f(j) = f(i)$ ; $||\mathbf{x}||^2 = \sum_i (x_i)^2$ which is just the Pythagorean theorem written out in index notation),
- also the ability to program in Julia and/or Python. Some of this material is reviewed in K&W Appendix A.

Many if not most of the present-day breakthroughs in machine learning, artificial intelligence, computer graphics, computer vision, scientific computing, etc. that seem to be transforming world culture actually rely on applications of mathematical numerical optimization (not to mention statistics and several other applied mathematical fields). Key to the very nature of numerical optimization is that it is quite mathematical. Correspondingly, …

From some more concrete points of view this class is fairly mathematical, as is the Instructor. If this is a problem for you, then *please do not take this class.* If you do not like mathematical abstraction, or if you can only really understand something through very concrete examples, then *this is not the right class for you.* On the other hand the Instructor will provide motivating examples in lectures, and the K&W book is quite code-oriented, and the B&C book is rich in engineering examples for possible projects. However, the lecture examples themselves will necessarily become more abstract as the course moves along.

**Workload**

As detailed above there will be about 4 homeworks that involve mathematical programming, and 3 quizzes that will be more mathematics-centric, a class participation grade, and a final group project.

**Academic integrity**

Academic integrity policy is, as usual for UCI, your responsibility to know, *and is important* in this class and to this instructor. Any external sources (cultural, human, or technological) used must be acknowledged and cited.

In assignments where AI tools such as Large Language Models are explicitly allowed, the full prompts used must be provided as if it were source code. If not explicitly allowed, such aids may not be used in the production of submitted work and such use is an academic integrity violation. If you had permission, and used them, but didn't learn much about the class content that way, don't submit that usage as completed homework.

Possible Student Outcomes

I don't really believe in "Student Outcomes" as some kind of promise of results from taking a course, because I believe instead that outcomes are proportionate to student inputs on a log-log scale, without much direct dependency on instructor input. In other words, student outcomes depend in practice mainly on student effort and enthusiasm for the subject. However, here are some outcomes that might plausibly result from investing effort in this class.

1. understanding the nature of reasons for a classification hierarchy of optimization problem and algorithm types;
2. understanding the role of condition number in estimating the computational cost of quadratic optimization by gradient descent and the cost savings available in conjugate gradient methods;
3. experience with at least one generic software framework for numerical optimization, including automatic differentiation;
4. knowledge of one or two representative optimization algorithms for each optimization problem type near the root of the foregoing hierarchy, including:
    1. zeroth order, first order, and second order methods for continuous local optimization;
    2. linear programming (lp);
    3. augmented lagrangians;
    4. some combinatorial optimization algorithm such as graph cut optimization;
5. knowledge of the KKT necessary and sufficient conditions for local solution of a variety of continuous optimization problem types;
6. understanding of the role of optimization and example problem formulations within several classic domains such as machine learning, computer vision, structural mechanics, and traffic flow;
7. knowledge of the definition and some of the uses of duality in constrained optimization;
8. awareness of highly scalable optimization algorithms that exploit parallel computing and/or multiple problem scales;
9. experience in creative application of optimization in a group final project.