1 The Math 693a “Standard Project”

Milestone #1

Figure: Code structure for the project. The top block contains all the drivers, common linear algebra solvers, stopping criteria checking, etc... The hess–fn–grad triplet defines the test function, its gradient and hessian; you will need one such “block” for each test function. The bottom four boxes define 4 different strategies.

The first milestone is to implement — from Dennis & Schnabel Appendix A — the driver block, the linesearch strategy, for one test function from Appendix B (see handout) — if you pick Rosenbrock then you can compare results with previous homework. This is STRONGLY recommended.

Milestone #2

The second milestone is to implement (at least) one additional strategy, and add (at least) one additional test function from Appendix B to your code-base. If you “only” implement one additional strategy, it is a BAD IDEA to select linesearchmod.

Initial implementation assumes:
analgrad=TRUE, analhess=TRUE, factsec=FALSE, cheapf="Does not matter"
**Milestone #3**

Add the following to your code base:

- `fdhessg` — Finite difference approximation to the Hessian using analytic gradient. (Executed when `analgrad=TRUE, analhess=FALSE, cheapf=TRUE`.)
- `fdjac` — The core call from `fdhessg`, note that `fvec` in the pseudo-code corresponds to your analytic gradient.
- `fdgrad` — Finite difference (forward) approximation to the gradient. (Executed when `analgrad=FALSE`.)
- `fdhessf` — Finite difference approximation to the Hessian using only function values. (Executed when `analgrad=FALSE, analhess=FALSE, cheapf=TRUE`.)

**Milestone #4**

Add something else to your framework: *e.g.* Conjugate Gradient as an overall solution strategy, or as Linear Systems solver “only”, or in the form of Steihaug’s method.

**Things to Explore (not an Exhaustive List)**

- **Test Functions**
  - (At least) 2 Different test functions (see Appendix B, and other resources)
  - Different starting points (see Appendix B)
  - Different dimensionality ($n$)

- **Finite Differencing**
  - Analytic everything (no FD)
  - Analytic gradient (`fdhessg+fdjac`)
  - Finite difference everything (`fdhessf+fdgrad`)
  - optimal and non-optimal $\epsilon$

**“Deliverables”**

- 10-15 minute presentation at the end of the semester
- Soft-copy of presentation
- Soft-copy of code
2 Other Project Options

Solve some other related (application) optimization problem that seems useful and exciting to you.
You can look deeper into something we have done, or use the class as a spring board for
exploration into the “next level” of strategies: e.g. Constrained problems — Linear and Quadratic
Programming, Interior Point methods, Penalty Methods, etc...

Some Examples of Past Projects

- Conjugate Gradient Based Methods
- Finding Steady State Vortex Solutions to the 2D Nonlinear Schrodinger Equation Using Opt-
  imization Algorithms
- Generating Time-Independent Solutions to the Gross-Pitaevskii Equation
- Image Restoration
- Influence of Step Size in Finite differencing of Hessian for Unconstrained Minimization Test
  Functions
- Nonlinear Parameter Fitting with Applications to Mathematical Biology
- Linear Least Squares – QR Decomposition; Conjugate Gradient
- Optimization of a Traveling Numeric Soliton Like Solution to the Nonlinear Schrodinger Equa-
  tion
- Optimization of Laguerre Type Orbital Basis Sets

Some Examples of Past Test Functions

- Branin Function
- Dixon Price Function
- Easom Function
- Extended Powell Function
- Helical Valley Function
- Powell Singular Function
- Trigonometric Function
- Wood Function
Some Examples of Past Implementation Environments

- C++
- Fortran
- Intel Math Kernel Library
- Mathematica
- Matlab
- Python w/NumPy Library