Tutorial 2: Unconstrained optimization

GIAN Short Course on Optimization: Applications, Algorithms, and Computation

Sven Leyffer

Argonne National Laboratory

September 12-24, 2016

Theory of Optimality Conditions

1. Prove Theorem 3.1.1 (necessary conditions for a local min).
   Hint: Consider $f(x)$ along lines s and use the 1D conditions.
2. Consider $f(x) = 2x_1^3 - 3x_1^2 - 6x_1x_2(x_1 - x_2 - 1)$. Plot $f(x)$ in domain $[-1, 1]^2$ using Matlab. Find its gradient and Hessian matrix, and find and classify all its stationary points.
3. Consider $f(x) = x^2$ with global minimum at $x = 0$. Show that iterates, $x^{(k+1)} = x^{(k)} + \alpha_k s^{(k)}$, defined $s^{(k)} = (-1)^{k+1}$ and $\alpha_k = 2 + 3/2^{k+1}$ starting from $x^{(0)} = 2$ satisfy $s^T g < 0$ and $f^{(k+1)} < f^{(k)}$. Plot iterates using Matlab, and show that Algorithm 6.1 does not converge to a stationary point.

Tutorial 2: Armijo Line Search

- Write a Matlab function that implements the Armijo line-search. Your function should be callable from other matlab rountines, e.g.

```
% ArmijoLineSearch.m Implementation of Armijo line search
function [alpha, j, info]
   = ArmijoLineSearch( x, dx, g, ObjFun, varargin )

% ... define the constants of the search
t     = 1.0;       % ... initial step size
beta  = 0.5 ;      % ... back-tracking factor
maxf  = 10;        % ... maximum number of function evals
sigma = 0.1;       % ... sufficient reduction parameter
```

  - Use the `Matlab` function `evalf` and function pointers to pass different `ObjFun` arguments.

Tutorial 2: Steepest Descend

- Test the line-search using the Powell function (`powell.m`), and the Rosenbrock function (`rosenbrock.m`), available on the web-site. In each case display the actual and predicted reduction for your step.
- Implement the steepest descend method with an Armijo line search in Matlab.
  - You can either implement your own line-search (preferred), or use mine.
  - Test the method on the two example above, and the chained Rosenbrock function.
  - Use Matlab's `contour` function to plot the contours (in 2D), and the progress of your method.
  - Starting points are $x = (-1.5; 0.5)$ for Rosenbrock, and $x = (1; 1)$ for Powell, and $x = (-1; -1; \ldots; -1)$ for chained Rosen.

Tutorial 2: Newton & Quasi-Newton Methods

- Inplement Newton's method with the Hessian modification.
  - Use your (or mine) steepest descend code as a starting point.
  - Use Matlab's eigenvalue functions, `eig`, to compute the eigenvalue.
  - Use Matlab's backslash operator to solve the Newton system.
  - Test the code again on our examples.
- Implement a quasi-Newton (or limited memomry BFGS) method.

Theory of Newton's Method

1. Show that Newton's method oscillates for
   min $f(x) = x^2 - x^4/4$.
1. Prove quadratic convergence of Newton's method (Theorem 4.1.1)
1. Show that (**??**) holds for a quadratic function.
1. Show that the rank-one formula terminates for a quadratic:
   - Show by induction that $H^{(k+1)}\gamma^{(j)} = \delta^{(j)}$ for all $j = 1, \ldots, k$.
   - Hence conclude that the method terminates after $n+1$ iterations.
1. Program the limited-memory BFGS method in Matlab.
1. Apply Newton's method to nonlinear least-squares:

$$\underset{x}{\text{minimize}} \;\; f(x) = \sum_{i=1}^{m} r_i(x)^2 = r(x)^T r(x) = \|r(x)\|_2^2.$$

   What happens, if $r_i(x)$ are linear? Can you propose a strategy for handling the case, where $\nabla^2 r_i(x)$ are bounded, and $r_i(x) \to 0$?