Line Search Methods for Unconstrained Optimisation

Lecture 8, Numerical Linear Algebra and Optimisation
Oxford University Computing Laboratory, MT 2007
Dr Raphael Hauser (hauser@comlab.ox.ac.uk)

The Generic Framework

For the purposes of this lecture we consider the unconstrained minimisation problem

$$\text{(UCM)} \quad \min_{x \in \mathbb{R}^n} f(x),$$

where $f \in C^1(\mathbb{R}^n, \mathbb{R})$ with Lipschitz continous gradient $g(x)$.

- In practice, these smoothness assumptions are sometimes violated, but the algorithms we will develop are still observed to work well.

- The algorithms we will construct have the common feature that, starting from an initial educated guess $x^0 \in \mathbb{R}^n$ for a solution of (UCM), a sequence of solutions $(x^k)_{\mathbb{N}} \subset \mathbb{R}^n$ is produced such that

  $$x^k \to x^* \in \mathbb{R}^n$$

  such that the first and second order necessary optimality conditions

  $$g(x^*) = 0,$$
  $$H(x^*) \succeq 0 \quad \text{(positive semidefiniteness)}$$

  are satisfied.

- We usually wish to make progress towards solving (UCM) in every iteration, that is, we will construct x^{k+1} so that

$$f(x^{k+1}) < f(x^k)$$

(descent methods).

- In practice we cannot usually compute x^* precisely (i.e., give a symbolic representation of it, see the LP lecture!), but we have to stop with a x^k sufficiently close to x^*.

- Optimality conditions are still useful, in that they serve as a stopping criterion when they are satisfied to within a predetermined error tolerance.

- Finally, we wish to construct $(x^k)_{\mathbb{N}}$ such that convergence to x^* takes place at a rapid rate, so that few iterations are needed until the stopping criterion is satisfied. This has to be counterbalanced with the computational cost per iteration, as there typically is a tradeoff

$$\text{faster convergence} \Leftrightarrow \text{higher computational cost per iteration.}$$

We write $f^k = f(x^k)$, $g^k = g(x^k)$, and $H^k = H(x^k)$.

Generic Line Search Method:

1. Pick an initial iterate x^0 by educated guess, set $k = 0$.

2. Until x^k has converged,

   i) Calculate a *search direction* p^k from x^k, ensuring that this direction is a *descent direction*, that is,

   $$[g^k]^\top p^k < 0 \text{ if } g^k \neq 0,$$

   so that for small enough steps away from x^k in the direction p^k the objective function will be reduced.

   ii) Calculate a suitable *steplength* $\alpha^k > 0$ so that

   $$f(x^k + \alpha^k p^k) < f^k.$$

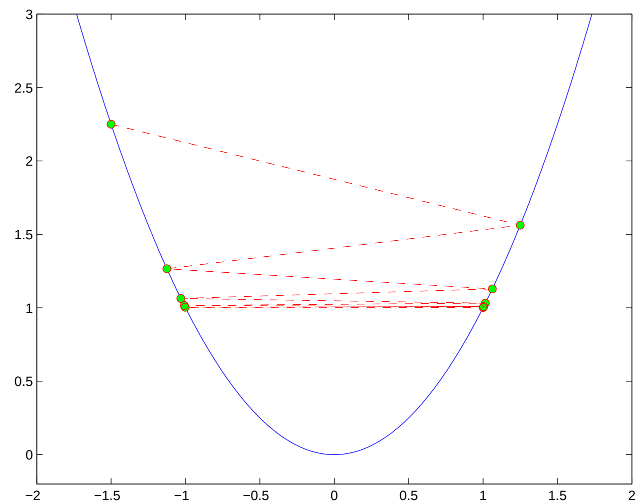
   The computation of α^k is called *line search*, and this is usually an inner iterative loop.

   iii) Set $x^{k+1} = x^k + \alpha^k p^k$.

Actual methods differ from one another in how steps i) and ii) are computed.

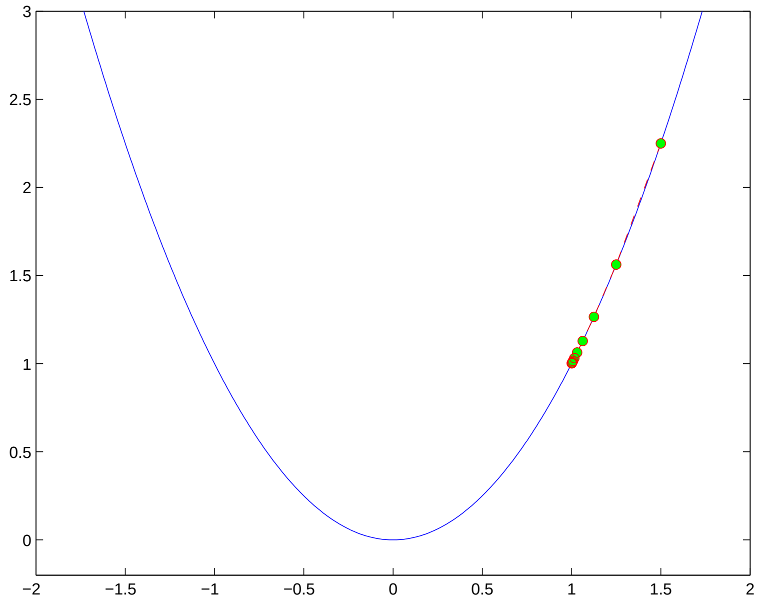
Computing a Step Length α^k

The challenges in finding a good α^k are both in avoiding that the step length is too long,



(the objective function $f(x) = x^2$ and the iterates $x^{k+1} = x^k + \alpha^k p^k$ generated by the descent directions $p^k = (-1)^{k+1}$ and steps $\alpha^k = 2+3/2^{k+1}$ from $x_0 = 2$)

or too short,



(the objective function $f(x) = x^2$ and the iterates $x^{k+1} = x^k + \alpha^k p^k$ generated by the descent directions $p^k = -1$ and steps $\alpha^k = 1/2^{k+1}$ from $x_0 = 2$).

Exact Line Search:

In early days, α^k was picked to minimize

$$\text{(ELS)} \quad \begin{aligned} &\min_{\alpha} f(x^k + \alpha p^k) \\ &\text{s.t. } \alpha \geq 0. \end{aligned}$$

Although usable, this method is not considered cost effective.

Inexact Line Search Methods:

- Formulate a criterion that assures that steps are neither too long nor too short.
- Pick a good initial stepsize.
- Construct sequence of updates that satisfy the above criterion after very few steps.

Backtracking Line Search:

1. Given $\alpha_{\text{init}} > 0$ (e.g., $\alpha_{\text{init}} = 1$), let $\alpha^{(0)} = \alpha_{\text{init}}$ and $l = 0$.

2. Until $f(x^k + \alpha^{(l)} p^k)$ "<" f^k,

   i) set $\alpha^{(l+1)} = \tau \alpha^{(l)}$, where $\tau \in (0, 1)$ is fixed (e.g., $\tau = \frac{1}{2}$),

   ii) increment l by 1.

3. Set $\alpha^k = \alpha^{(l)}$.

This method prevents the step from getting too small, but it does not prevent steps that are too long relative to the decrease in f.

To improve the method, we need to tighten the requirement

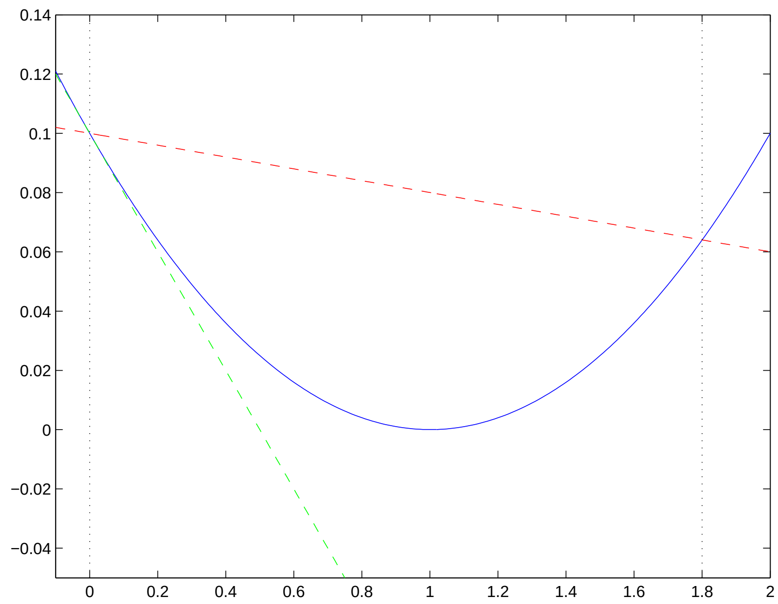
$$f(x^k + \alpha^{(l)} p^k) \text{ ``<'' } f^k.$$

To prevent long steps relative to the decrease in f, we require the *Armijo condition*

$$f(x^k + \alpha^k p^k) \leq f(x^k) + \alpha^k \beta \cdot [g^k]^\top p^k$$

for some fixed $\beta \in (0, 1)$ (e.g., $\beta = 0.1$ or even $\beta = 0.0001$).

That is to say, we require that the achieved reduction if f be at least a fixed fraction β of the reduction promised by the first-oder Taylor approximation of f at x^k.



Backtracking-Armijo Line Search:

1. Given $\alpha_{\text{init}} > 0$ (e.g., $\alpha_{\text{init}} = 1$), let $\alpha^{(0)} = \alpha_{\text{init}}$ and $l = 0$.

2. Until $f(x^k + \alpha^{(l)} p^k) \le f(x^k) + \alpha^{(l)} \beta \cdot [g^k]^\top p^k$,

   i) set $\alpha^{(l+1)} = \tau \alpha^{(l)}$, where $\tau \in (0, 1)$ is fixed (e.g., $\tau = \frac{1}{2}$),

   ii) increment l by 1.

3. Set $\alpha^k = \alpha^{(l)}$.

Theorem 1 (Termination of Backtracking-Armijo). *Let $f \in C^1$ with gradient $g(x)$ that is Lipschitz continuous with constant γ^k at x^k, and let p^k be a descent direction at x^k. Then, for fixed $\beta \in (0,1)$,*

i) the Armijo condition $f(x^k + \alpha p^k) \le f^k + \alpha\beta \cdot [g^k]^\top p^k$ is satisfied for all $\alpha \in [0, \alpha^k_{\mathrm{max}}]$, where

$$\alpha^k_{\mathrm{max}} = \frac{2(\beta - 1)[g^k]^\top p^k}{\gamma^k \|p^k\|_2^2},$$

ii) and furthermore, for fixed $\tau \in (0,1)$ the stepsize generated by the backtracking-Armijo line search terminates with

$$\alpha^k \ge \min\left(\alpha_{\mathrm{init}}, \frac{2\tau(\beta - 1)[g^k]^\top p^k}{\gamma^k \|p^k\|_2^2}\right).$$

We remark that in practice γ^k is not known. Therefore, we cannot simply compute α^k_{max} and α^k via the explicit formulas given by the theorem, and we still need the algorithm on the previous slide.

Theorem 2 (Convergence of Generic LSM with B-A Steps). *Let the gradient g of $f \in C^1$ be uniformly Lipschitz continuous on $\mathbb{R}^n$. Then, for the iterates generated by the Generic Line Search Method with Backtracking-Armijo step lengths, one of the following situations occurs,*

i) $g^k = 0$ *for some finite* k,

ii) $\lim_{k\to\infty} f^k = -\infty$,

iii) $\lim_{k\to\infty} \min\left(|[g^k]^\top p^k|, \frac{|[g^k]^\top p^k|}{\|p^k\|_2}\right) = 0.$

Computing a Search Direction p^k

Method of Steepest Descent:

The most straight-forward choice of a search direction, $p^k = -g^k$, is called *steepest-descent* direction.

- p^k is a descent direction.

- p^k solves the problem

$$\begin{aligned} \min p \in \mathbb{R}^n \ & \mathrm{m}_k^L(x^k + p) = f^k + [g^k]^\mathsf{T} p \\ \text{s.t.} \ & \|p\|_2 = \|g^k\|_2. \end{aligned}$$

- p^k is cheap to compute.

Any method that uses the steepest-descent direction as a search direction is a *method of steepest descent*.

Intuitively, it would seem that p^k is the best search-direction one can find. If that were true then much of optimisation theory would not exist!

Theorem 3 (Global Convergence of Steepest Descent). *Let the gradient g of $f \in C^1$ be uniformly Lipschitz continuous on $\mathbb{R}^n$. Then, for the iterates generated by the Generic LSM with B-A steps and steepest-descent search directions, one of the following situations occurs,*

i) $g^k = 0$ *for some finite* k,

ii) $\lim_{k\to\infty} f^k = -\infty$,

iii) $\lim_{k\to\infty} g^k = 0$.

Advantages and disadvantages of steepest descent:

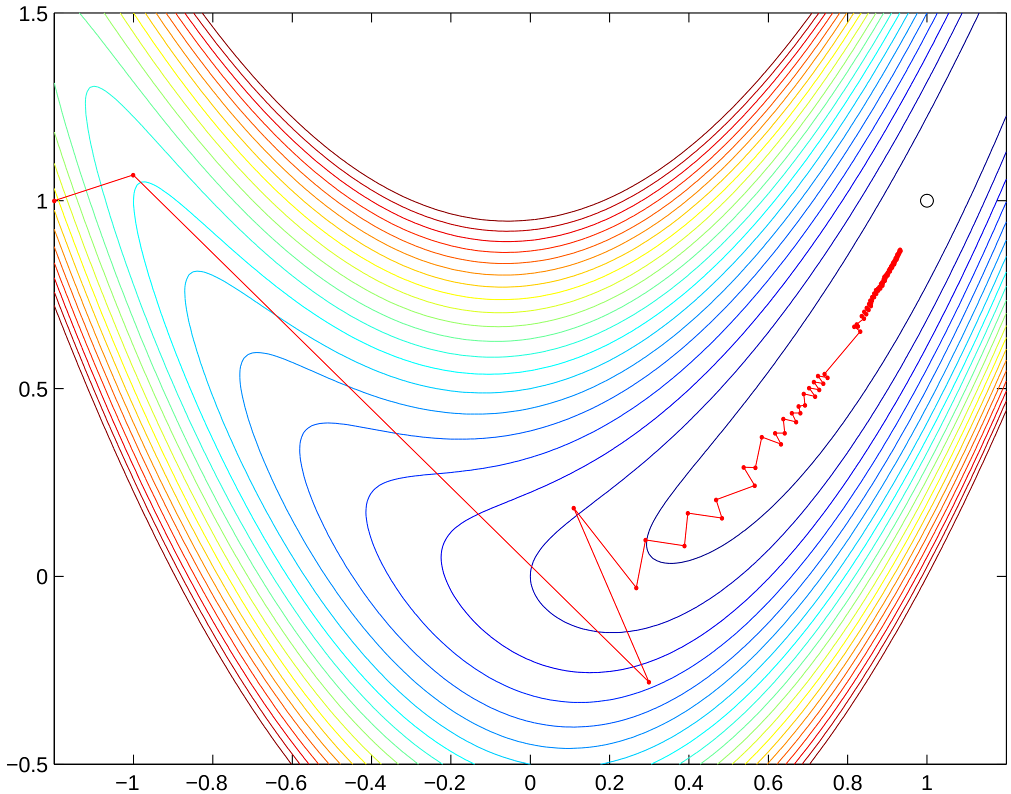
- $\oplus$ Globally convergent (converges to a local minimiser from any starting point x^0).

- $\oplus$ Many other methods switch to steepest descent when they do not make sufficient progress.

- $\ominus$ Not scale invariant (changing the inner product on $\mathbb{R}^n$ changes the notion of gradient!).

- $\ominus$ Convergence is usually very (very!) slow (linear).

- $\ominus$ Numerically, it is often not convergent at all.



Contours for the objective function $f(x,y) = 10(y - x^2)^2 + (x-1)^2$ (Rosenbrock function), and the iterates generated by the generic line search steepest-descent method.

More General Descent Methods:

Let B^k be a symmetric, positive definite matrix, and define the search direction p^k as the solution to the linear system

$$B^k p^k = -g^k$$

- p^k is a descent direction, since

$$[g^k]^\top p^k = -[g^k]^\top [B^k]^{-1} g^k < 0.$$

- p^k solves the problem

$$\min_{p \in \mathbb{R}^n} \mathrm{m}_k^Q(x^k + p) = f^k + [g^k]^\top p + \frac{1}{2} p^\top B^k p.$$

- p^k corresponds to the steepest descent direction if the norm

$$\|x\|_{B^k} := \sqrt{x^\mathsf{T} B^k x}$$

is used on $\mathbb{R}^n$ instead of the canonical Euclidean norm. This change of metric can be seen as preconditioning that can be chosen so as to speed up the steepest descent method.

- If the Hessian H^k of f at x^k is positive definite, and $B^k = H^k$, this is *Newton's method*.

- If B^k changes at every iterate x^k, a method based on the search direction p^k is called *variable metric* method. In particular, Newton's method is a variable metric method.

Theorem 4 (Global Convergence of More General Descent Direction Methods). *Let the gradient g of $f \in C^1$ be uniformly Lipschitz continuous on $\mathbb{R}^n$. Then, for the iterates generated by the Generic LSM with B-A steps and search directions defined by $B^k p^k = -g^k$, one of the following situations occurs,*

i) $g^k = 0$ *for some finite* k*,*

ii) $\lim_{k\to\infty} f^k = -\infty$,

iii) $\lim_{k\to\infty} g^k = 0$,

provided that the eigenvalues of B^k are uniformly bounded above, and uniformly bounded away from zero.

Theorem 5 (Local Convergence of Newton's Method)**.** *Let the Hessian H of $f \in C^2$ be uniformly Lipschitz continuous on $\mathbb{R}^n$. Let iterates x^k be generated via the Generic LSM with B-A steps using $\alpha_{\text{init}} = 1$ and $\beta < \frac{1}{2}$, and using the Newton search direction n^k, defined by $H^k n^k = -g^k$. If $(x^k)_{\mathbb{N}}$ has an accumulation point x^* where $H(x^*) \succ 0$ (positive definite) then*

i) $\alpha^k = 1$ *for all k large enough,*

ii) $\lim_{k\to\infty} x^k = x^*$,

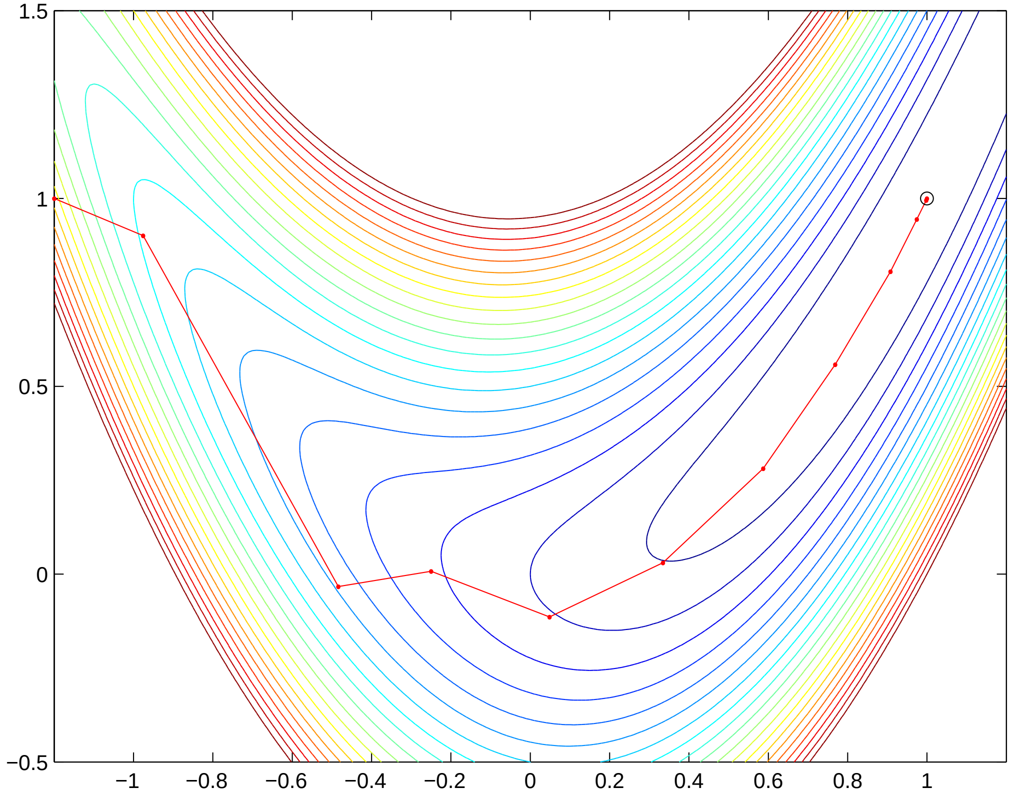
iii) the sequence converges Q-quadratically, that is, there exists $\kappa > 0$ such that

$$\lim_{k\to\infty} \frac{\|x^{k+1} - x^*\|}{\|x^k - x^*\|^2} \leq \kappa.$$

The mechanism that makes Theorem 5 work is that once the sequence $(x^k)_{\mathbb{N}}$ enters a certain domain of attraction of x^*, it cannot escape again and quadratic convergence to x^* commences.

Note that this is only a local convergence result, that is, Newton's method is not guaranteed to converge to a local minimiser from all starting points.

The fast convergence of Newton's method becomes apparent when we apply it to the Rosenbrock function:



Contours for the objective function $f(x,y) = 10(y-x^2)^2 + (x-1)^2$, and the iterates generated by the Generic Linesearch Newton method.

Modified Newton Methods:

The use of $B^k = H^k$ makes only sense at iterates x^k where $H^k \succ 0$. Since this is usually not guaranteed to always be the case, we modify the method as follows,

- Choose $M^k \succeq 0$ so that $H^k + M^k$ is "sufficiently" positive definite, with $M^k = 0$ if H^k itself is sufficiently positive definite.

- Set $B^k = H^k + M^k$ and solve $B^k p^k = -g^k$.

The *regularisation term* M^k is typically chosen as one of the following,

- If H^k has the spectral decomposition $H^k = Q^k \Lambda^k [Q^k]^\top$, then
$$H^k + M^k = Q^k \max(\varepsilon \mathrm{I}, |D^k|)[Q^k]^\top.$$

- $M^k = \max(0, -\lambda_{\text{min}}(H^k))\, \mathrm{I}$.

- Modified Cholesky method:

  1. Compute a factorisation $PH^kP^\top = LBL^\top$, where P is a permutation matrix, L a unit lower triangular matrix, and B a block diagonal matrix with blocks of size 1 or 2.

  2. Choose a matrix F such that $B + F$ is sufficiently positive definite.

  3. Let $H^k + M^k = P^\top L(B + F)L^\top P$.

Other Modifications of Newton's Method:

1. Build a cheap approximation B^k to H^k:

   - Quasi-Newton approximation (BFGS, SR1, etc.),

   - or use finite-difference approximation.

2. Instead of solving $B^k p^k = -g^k$ for p^k, if $B^k \succ 0$ approximately solve the convex quadratic programming problem

$$\text{(QP)} \quad p^k \approx \arg\min_{p \in \mathbb{R}^n} f^k + p^\mathsf{T} g^k + \frac{1}{2} p^\mathsf{T} B p.$$

The conjugate gradient method is a good solver for step 2:

1. Set $p^{(0)} = 0$, $g^{(0)} = g^k$, $d^{(0)} = -g^k$, and $i = 0$.

2. Until $g^{(i)}$ is sufficiently small or $i = n$, repeat

   i) $\alpha^{(i)} = \frac{\|g^{(i)}\|_2^2}{[d^{(i)}]^\top B^k d^{(i)}}$,

   ii) $p^{(i+1)} = p^{(i)} + \alpha^{(i)} d^{(i)}$,

   iii) $g^{(i+1)} = g^{(i)} + \alpha^{(i)} B^k d^{(i)}$,

   iv) $\beta^{(i)} = \frac{\|g^{(i+1)}\|_2^2}{\|g^{(i)}\|_2^2}$,

   v) $d^{(i+1)} = -g^{(i+1)} + \beta^{(i)} d^{(i)}$,

   vi) increment i by 1.

3. Output $p^k \approx p^{(i)}$.

Important features of the conjugate gradient method:

- $[g^k]^\top p^{(i)} < 0$ for all i, that is, the algorithm always stops with a descent direction as an approximation to p^k.

- Each iteration is cheap, as it only requires the computation of matrix-vector and vector-vector products.

- Usually, $p^{(i)}$ is a good approximation of p^k well before $i = n$.