# Nonlinear Optimization: Algorithms 1: Unconstrained Optimization

***INSEAD, Spring 2006***

Jean-Philippe Vert

Ecole des Mines de Paris

Jean-Philippe.Vert@mines.org

# Outline

- Descent methods
- Line search
- Gradient descent method
- Steepest descent method
- Newton's method
- Conjugate gradient method
- Quasi-Newton's methods

# Descent Methods

# Unconstrained optimization

- We consider the problem:

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

where $f$ is supposed to be *continuously differentiable*.

- We know that is $x^*$ is a local minimum it must satisfy (like all stationary points):

$$\nabla f\left(x^*\right) = 0 \, .$$

- In most cases this equation can not be solved analytically

# Iterative methods

- In practice we often use an *iterative algorithm* that computes a sequence of points:

$$x^{(0)}, x^{(1)}, \ldots\ldots, \in \mathbb{R}^n$$

with

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

- The algorithm typically stops when $\nabla f\left(x^{(k)}\right) < \epsilon$ for pre-defined $\epsilon$.
- No guarantee to find a global minimum..

# Strongly convex functions

Suppose that $f$ is *strongly convex*, i.e., there exists $m > 0$ with

$$\nabla^2 f(x) \succeq mI \,, \quad \forall x \in \mathbb{R}^n \,.$$

In that case we have the following bound:

$$f(x) - f^* \leq \frac{1}{2m} \| \nabla f(x) \|^2 \,,$$

and

$$\| x - x^* \| \leq \frac{1}{2m} \| \nabla f(x) \| \,,$$

yielding useful *stopping criteria* is $m$ is known, e.g.:

$$\| \nabla f(x) \| \leq \sqrt{2m\epsilon} \implies f(x) - f^* \leq \epsilon \,.$$

# Proofs

For any $x, y$, there exists a $z$ such that:

$$\begin{aligned} f(y) &= f(x) + \nabla f(x)^\top (y - x) + \frac{1}{2}(y - x)^\top \nabla^2 f(z)(y - x) \\ &\geq f(x) + \nabla f(x)^\top (y - x) + \frac{m}{2} \| y - x \|^2 \,. \end{aligned}$$

For fixed $x$, the r.h.s. is a convex quadratic function of $y$ that can be optimized w.r.t. $y$, yielding $\bar{y} = x - (1/m)\nabla f(x)$ and:

$$f(y) \geq f(x) - \frac{1}{2m} \| \nabla f(x) \|^2 \,, \quad \forall y \in \mathbb{R}^n \,.$$

$$\Longrightarrow \; f^* \geq f(x) - \frac{1}{2m} \| \nabla f(x) \|^2 \,.$$

# Proofs (cont.)

Applying the first upper bound to $y = x^*$, we obtain with Cauchy-Schwarz:

$$\begin{aligned} f^* = f(x^*) &\geq f(x) + \nabla f(x)^\top (x^* - x) + \frac{m}{2} \| x^* - x \|^2 \\ &\geq f(x) - \| \nabla f(x) \| \| x^* - x \| + \frac{m}{2} \| x^* - x \|^2 . \end{aligned}$$

Since $f(x) \geq f^*$ we must have

$$-\| \nabla f(x) \| \| x^* - x \| + \frac{m}{2} \| x^* - x \|^2 \leq 0$$

$$\Longrightarrow \| x - x^* \| \leq \frac{1}{2m} \| f(x) \| . \quad \square$$

# Descent method

We consider iterative algorithms which produce points

$$x^{(k+1)} = x^{(k)} + t^{(k)} \Delta x^{(k)} , \quad \textit{with} \quad f\left(x^{(k+1)}\right) < f\left(x^{(k)}\right)$$

- $\Delta x^{(k)} \in \mathbb{R}^n$ is the *step direction* or *search direction*.
- $t^{(k)}$ is the *step size* or *step length*.

A safe choice for the search direction is to take a *descent direction*, i.e., which satisfy:

$$\nabla f\left(x^{(k)}\right) \Delta x^{(k)} < 0$$

# General descent method

- *given* a starting point $x \in \mathbb{R}^n$.
- *repeat*
  1. Determine a descent direction $\Delta x$.
  2. Line search: choose a step size $t > 0$
  3. Update: $x := x + t\Delta x$.
- *until* stopping criterion is satisfied.

# Questions

- How to choose the descent direction?
  - Gradient method
  - Newton's method
  - Conjugate gradient method
  - Quasi-gradient methods
- How to choose the step size? (line search)

Different methods have different complexities, and different speeds of convergence...

# Line search

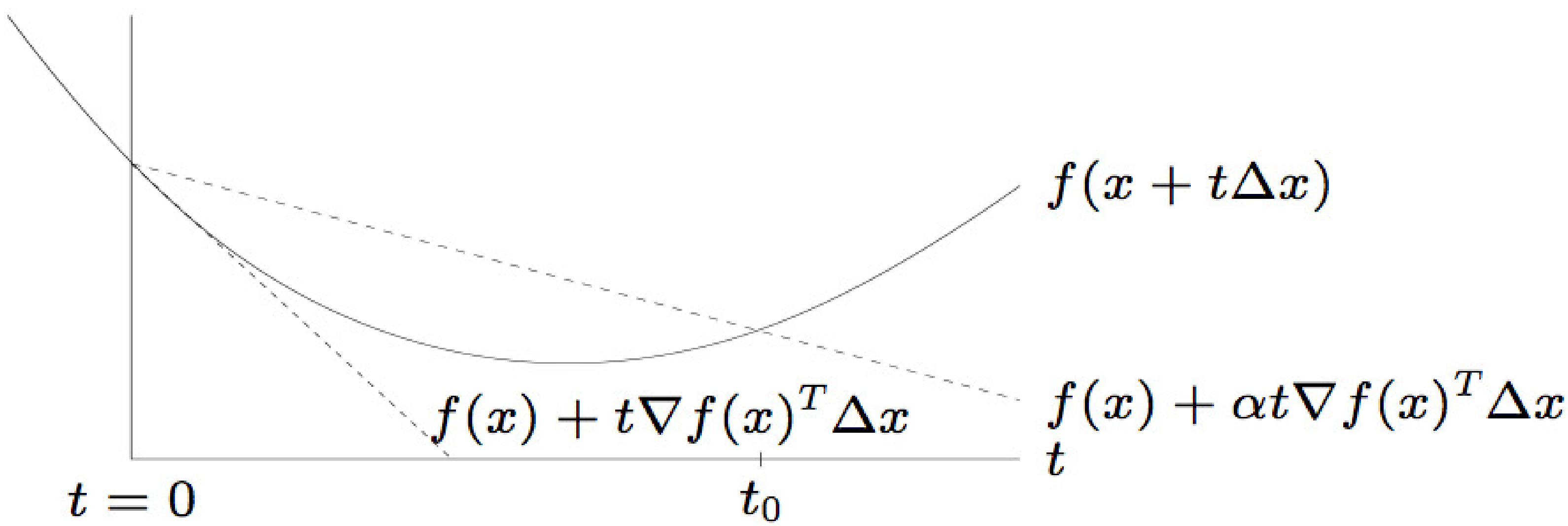
# Minimization rule

- Choose $t^{(k)}$ such that
$$f\left(x^{(k)} + t^{(k)}\Delta x^{(k)}\right) = \min_{t\geq 0} f\left(x^{(k)} + t\Delta x^{(k)}\right) .$$
- Useful when the cost of the minimization to find the step size is low compared to the cost of computing the search direction (e.g., analytic expression for the minimum).
- *Limited minimization rule*: same as above with some restriction on the step size (useful is the line search is done computationally):
$$f\left(x^{(k)} + t^{(k)}\Delta x^{(k)}\right) = \min_{0\leq t\leq s} f\left(x^{(k)} + t\Delta x^{(k)}\right) .$$

# Backtracking line search

- *Backtracking line search*, aka *Armijo rule*
- Given a descent direction $\Delta x$ for $f$ at $x$, and $\alpha \in (0, 0.5)$, $\beta \in (0, 1)$.
- Starting at $t = 1$, repeat $t := \beta t$ until

$$f\left(x + t\Delta x\right) < f(x) + \alpha t \nabla f(x)^\top \Delta x$$



# Alternative methods

- *Constant stepsize*:

$$t^{(k)} = cte \ .$$

- *Diminishing stepsize*:

$$t^{(k)} \to 0 \ ,$$

but satisfies the infinite travel condition:

$$\sum_{k=1}^{\infty} t^{(k)} = \infty \ .$$

# Line search: summary

- Exact minimization is only possible in *particular cases*.
- For most descent methods, the optimal point is *not required* in the line search.
- *Backtracking* is easily implemented and works well in practice

# Gradient descent method

# Gradient descent method

A natural choice for the search direction is the negative gradient

$$\Delta x = -\nabla f(x) \ ,$$

The resulting algorithm is called the *gradient algorithm* or *gradient descent method*:

- *given* a starting point $x \in \mathbb{R}^n$.
- *repeat*
  1. $\Delta x = -\nabla f(x)$.
  2. Line search: choose a step size $t > 0$ via exact or backtracking line search
  3. Update: $x := x + t\Delta x$.
- *until* stopping criterion is satisfied, e.g., $\| \nabla f(x) \|_2 \leq \eta$.

# Convergence analysis

For $f$ strictly convex, let $m, M$ s.t:

$$mI \preceq \nabla^2 f(x) \preceq MI \,, \quad \forall x \in \mathbb{R}^n \,.$$

For the exact line search method we can show that for any $k$,

$$f\left(x^{(k+1)}\right) - f^* \leq \left(1 - \frac{m}{M}\right)\left(f\left(x^{(k)}\right) - f^*\right) \,.$$

This shows that $f\left(x^{(k)}\right) \to f^*$ for $k \to \infty$. The convergence is geometric, but can be very slow if the *conditioning number* $m/M$ is small.

# Proof (for exact line search)

For a fixed $x$, let $g(t) = f(x - t\nabla f(x))$. From $\nabla^2 f(x) \preceq MI$ we deduce, using an upper bound of the second-order Taylor expansion:

$$g(t) \leq f(x) - t\| \nabla f(x) \|_2^2 + \frac{Mt^2}{2} \| \nabla f(x) \|^2$$

Minimizing both sides w.r.t. $t$, and taking $x = x^{(k)}$ we obtain:

$$f\left(x^{(k+1)}\right) - f^* \leq f\left(x^{(k+1)}\right) - f^* - \frac{1}{2M} \| \nabla f(x) \|^2 \,.$$

Using finally $\| \nabla f(x) \|^2 \geq 2m\left(f(x) - f^*\right)$, we get:

$$f\left(x^{(k+1)}\right) - f^* \leq \left(1 - \frac{m}{M}\right)\left(f\left(x^{(k}\right) - f^*\right) \,.$$

See B&V p.468 for the case of backtracking line search. □

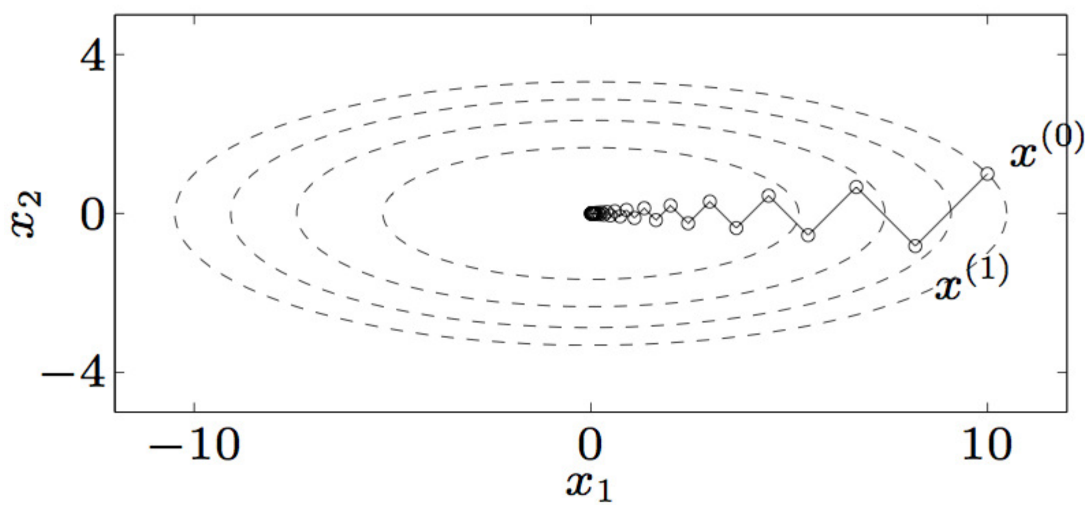
# Example 1: Quadratic problem in $\mathbb{R}^2$

$$f(x) = \frac{1}{2}\left(x_1^2 + \gamma x_2^2\right) \ ,$$

with exact line search, starting at $x^{(0)} = (\gamma, 1)$:

$$x_1^{(k)} = \gamma\left(\frac{\gamma-1}{\gamma+1}\right)^k \ , \quad x_2^{(k)} = \gamma\left(-\frac{\gamma-1}{\gamma+1}\right)^k$$

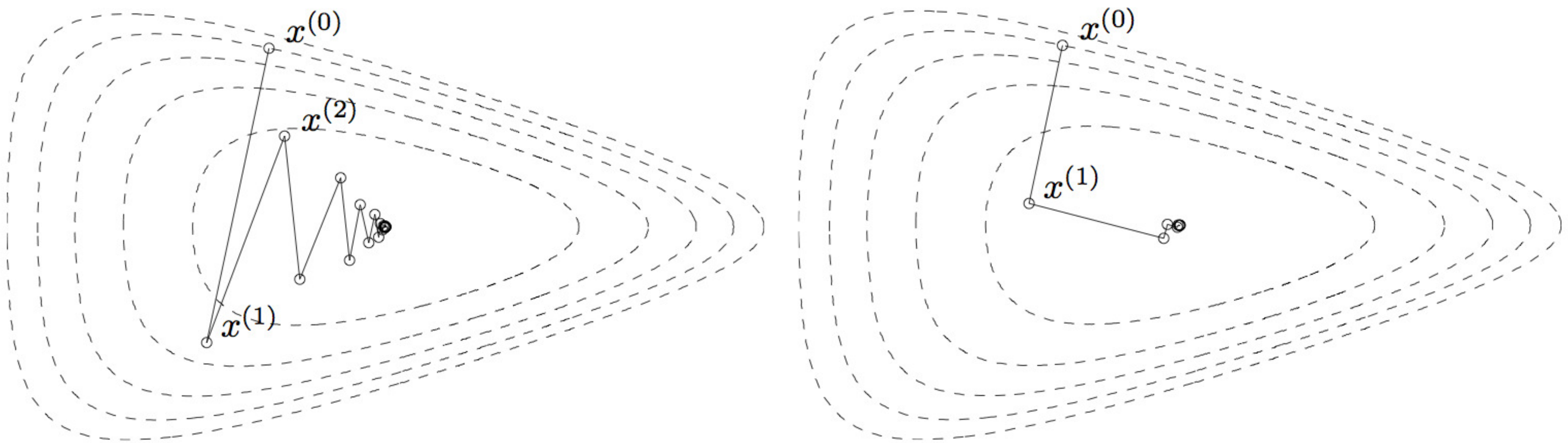
- *very slow* if $\gamma \gg 1$ or $\gamma \ll 1$.
- Example for $\gamma = 10$:



# Example 2: Non-quadratic problem

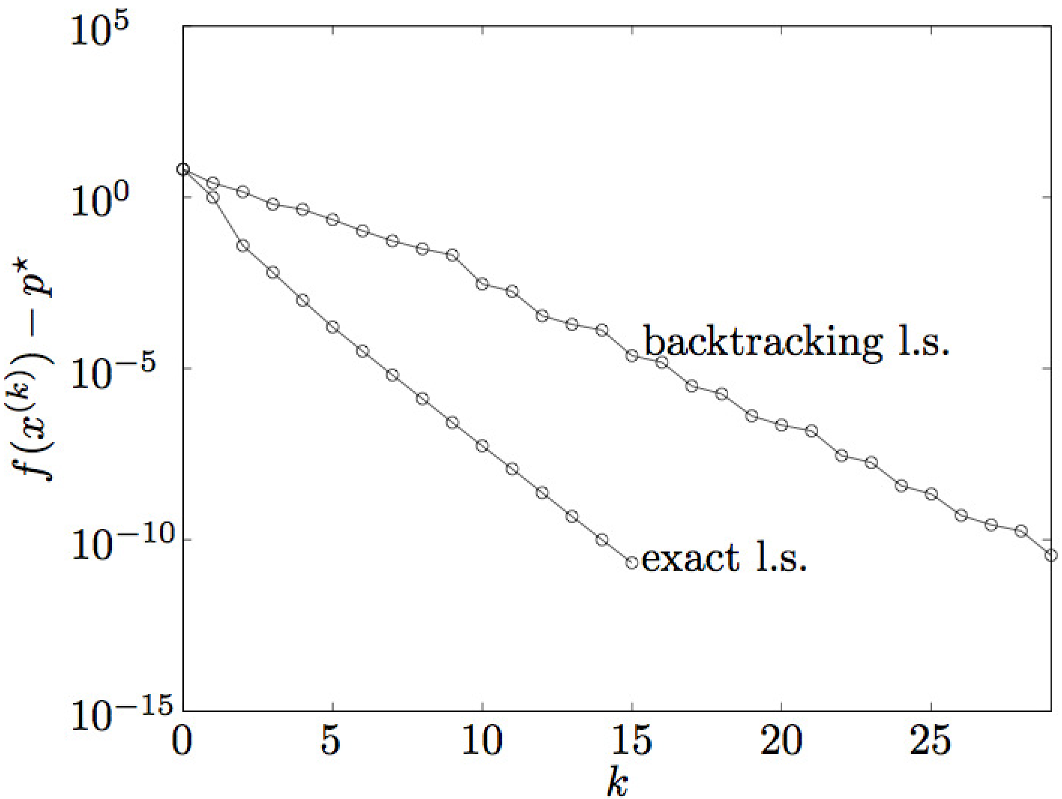
$$f(x) = e^{x_1+3x_2-0.1} + e^{x_1-3x_2-0.1} + e^{-x_1-0.1}$$

Backtracking ($\alpha = 0.1$, $\beta = 0.7$) vs. exact search:



# Example 2: speed of convergence

$$f(x) = e^{x_1+3x_2-0.1} + e^{x_1-3x_2-0.1} + e^{-x_1-0.1}$$




"*Linear convergence*", i.e., straight line on a semilog plot.

# Gradient descent summary

- The gradient method often exhibits *linear convergence*, i.e., $f\left(x^{(k)}\right) - f^*$ converges to $0$ geometrically.
- The choice of backtracking parameters has a noticeable but not dramatic effect on the convergence. $\alpha = 0.2 - 0.5$ and $\beta = 0.5$ is a safe default choice. Exact line search is painful to implement and has no dramatic effect.
- The convergence rate depends greatly on the *condition number of the Hessian*. When the condition number is 1000 or more, the gradient method is so slow that it is useless in practice.
- *Very simple, but rarely used in practice due to slow convergence.*

# Steepest descent method

# Motivations

The first-order Taylor approximation around $x$ is:

$$f(x+v) \sim f(x) + \nabla f(x)^\top v \ .$$

A good descent direction $v$ should make the term $\nabla f(x)^\top v$ as small as possible. Restricting $x$ to be in a unit ball we obtain a *normalized steepest descent direction:*

$$\Delta x = \arg\min \left\{ \nabla f(x)^\top v \mid \| v \| \leq 1 \right\} \ ,$$

i.e., the direction in the unit ball of $\| . \|$ that extends furthest in the direction of $-\nabla f(x)$.

# Euclidean norm

The solution of

$$\min\left\{\nabla f(x)^\top v \mid \| v \|_2 \leq 1\right\}$$

is easily obtained by taking:

$$v = \frac{\nabla f(x)}{\| \nabla f(x) \|_2} .$$

Therefore *gradient descent method is the steepest descent method for the Euclidean norm*.

# Quadratic norm

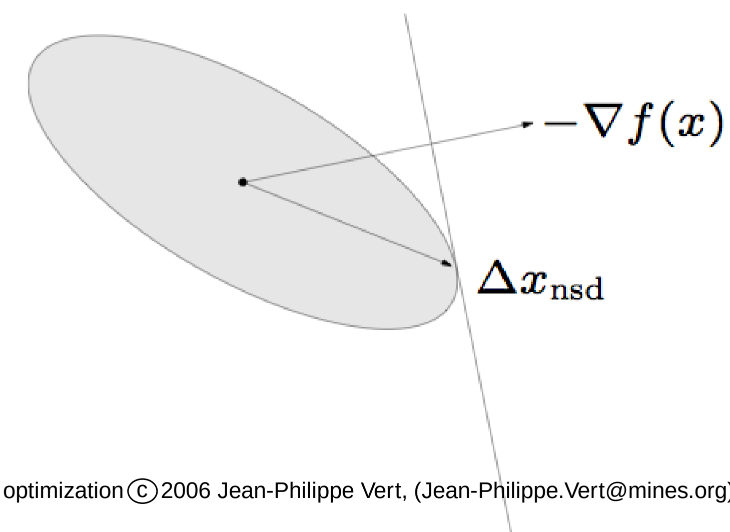
We consider the quadratic norm defined for $P \succ 0$ by:

$$\| x \|_P = \left( x^\top P x \right)^{\frac{1}{2}} = \| P^{\frac{1}{2}} x \|_2 \,.$$

The normalized steepest descent direction is given by:

$$v = \frac{-P^{-1} \nabla f(x)}{\| P^{-1} \nabla f(x) \|_P} \,.$$

The steepest descent method in the quadratic norm $\| . \|_P$ can be thought of as the gradient method applied to the problem after the change of coordinates $x \mapsto P^{\frac{1}{2}} x$.



# $l_1$ norm

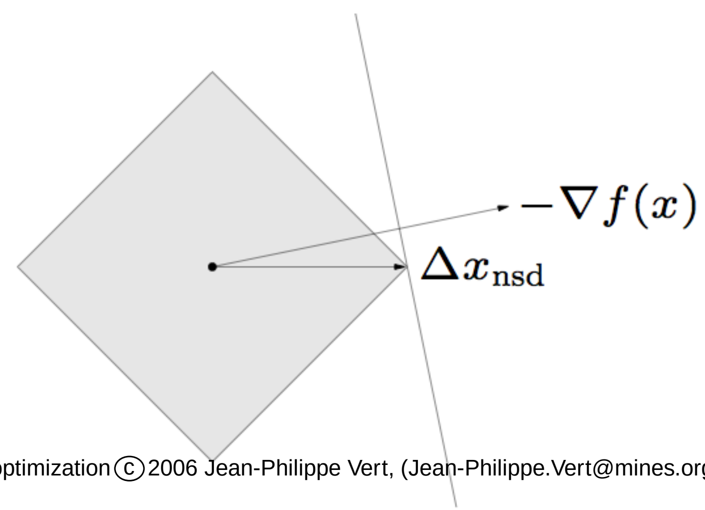
We consider the *$l_1$ norm*:

$$\| x \|_1 = \sum_{i=1}^{n} | x_i | \ .$$

The normalized steepest descent direction is given by:

$$v = -\text{sign}\left(\frac{\partial f(x)}{\partial x_i}\right) e_i \ , \quad \left|\frac{\partial f(x)}{\partial x_i}\right| = \max_j \left|\frac{\partial f(x)}{\partial x_i}\right| \ .$$

At each iteration we select a component of $\nabla f(x)$ with maximum absolute value, and then decrease or increase the corresponding component of $x$. This is sometimes called *coordinate-descent algorithm*.



# Convergence analysis

Convergence properties are similar to the gradient method:

$$f\left(x^{(k+1)}\right) - f^* \leq c\left(f\left(x^{(k)}\right) - f^*\right) .$$

where $c$ depends on the norm chosen. We therefore have *linear convergence* for all steepest descent method.

*Proof:* all norm are equivalent so there exists a scalar $\gamma$ such that $\| x \| \geq \gamma \| x \|_2$. Plug this into the proof of for the gradient descent (see B&V p.479).

# Choice of the norm

- The choice of the norm can have a *dramatic effect* on the *convergence rate* (changes the conditioning number).
- For the quadratic $P$ norm, the smallest condition number is obtained with

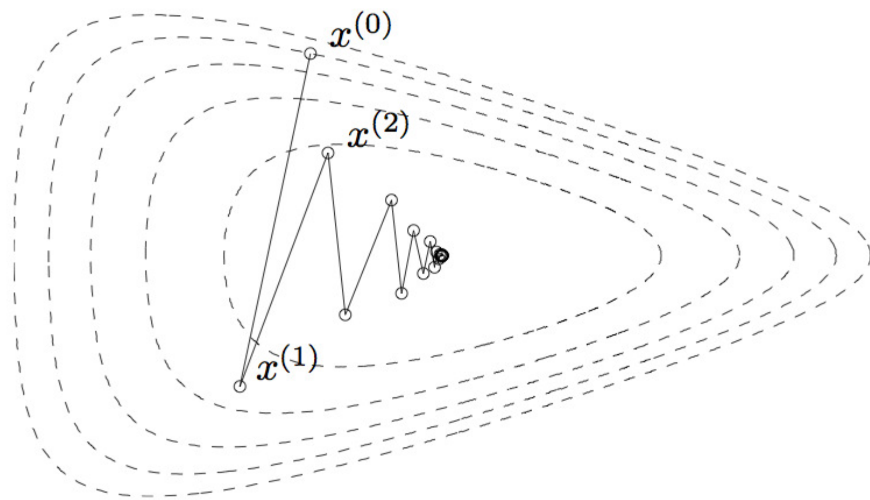
$$P = \nabla^2 f(x) \; .$$

  because the Hessian after the transformation $x \mapsto P^{\frac{1}{2}} x$ is $I$.
- In practice, steepest descent with quadratic $P$ norm works well in cases where we can identify a matrix $P$ for which the transformed problem has *moderate condition number*.

# Example

$$f(x) = e^{x_1+3x_2-0.1} + e^{x_1-3x_2-0.1} + e^{-x_1-0.1}$$

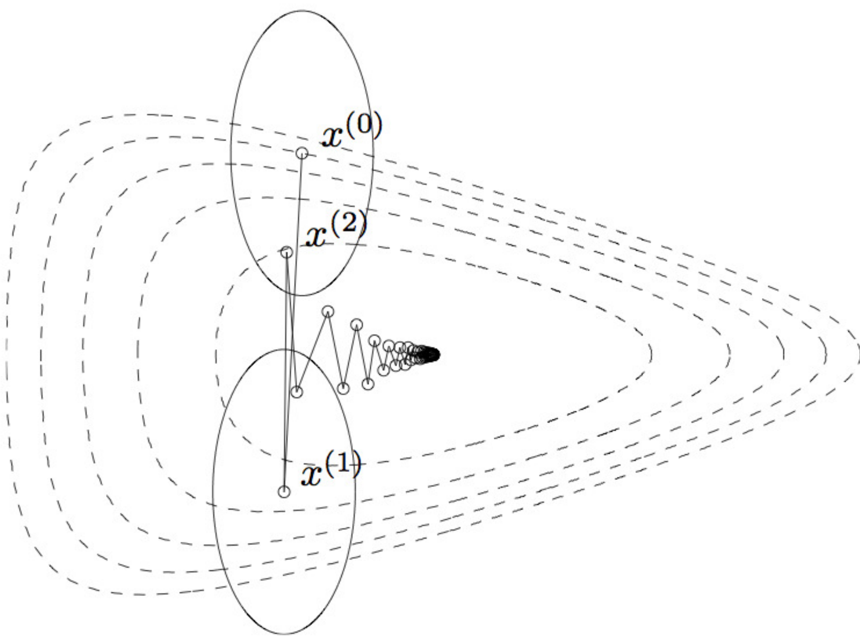
Backtracking ($\alpha = 0.1$, $\beta = 0.7$) for the gradient method:

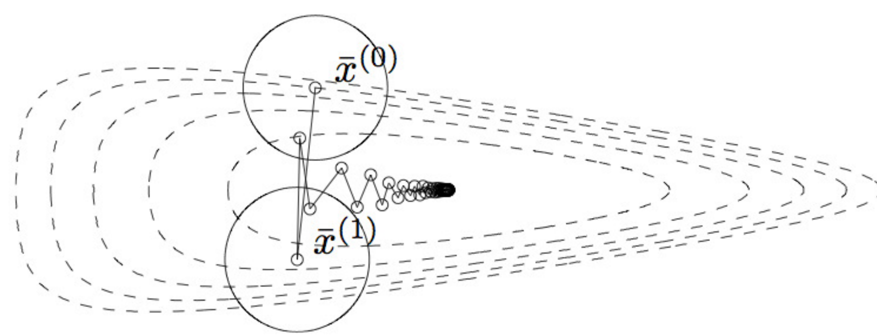



Let us study steepest descent methods with quadratic $P$ norm for different $P$'s in this case.

# Example: bad choice

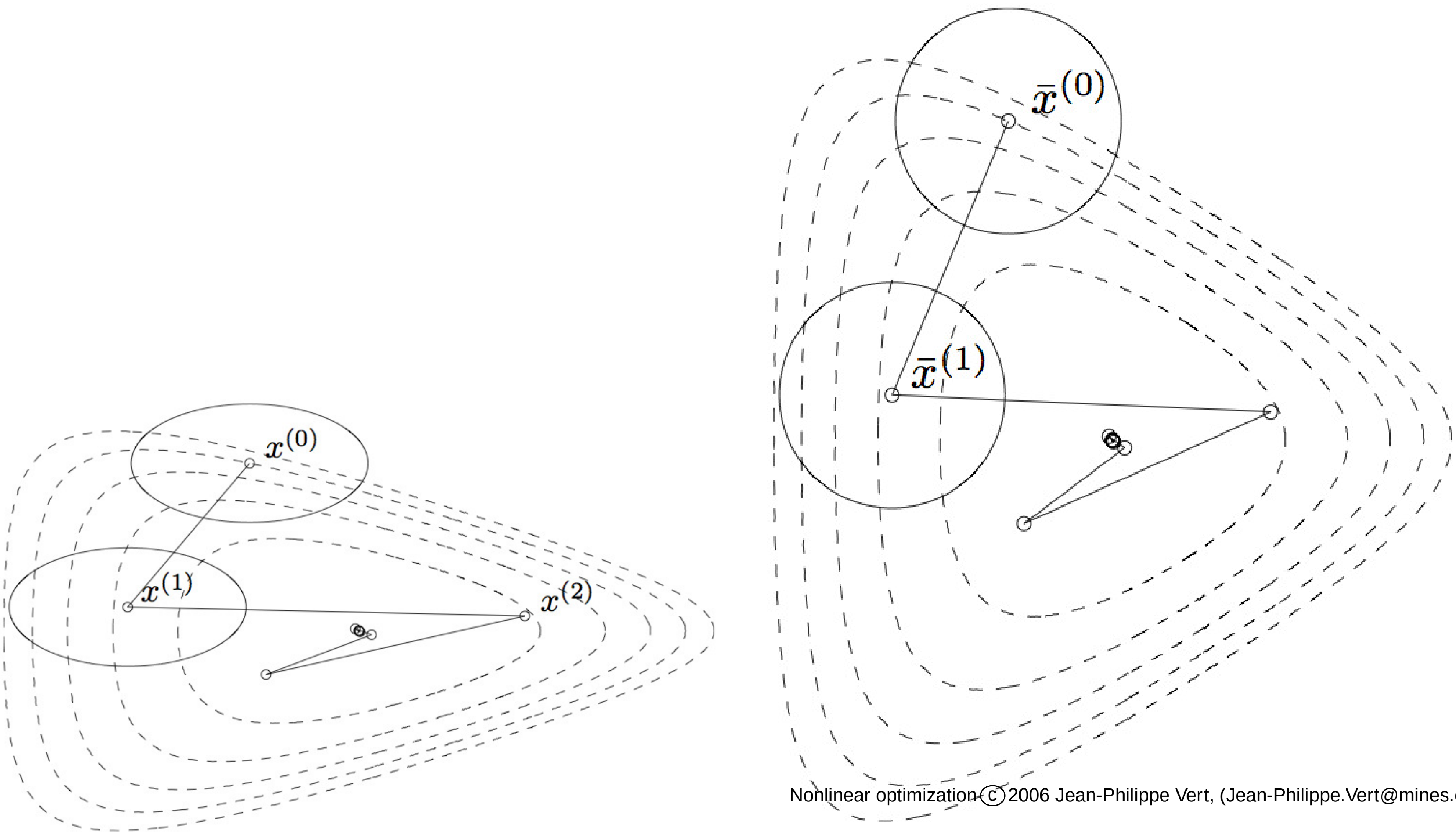
$$P = \begin{pmatrix} 8 & 0 \\ 0 & 2 \end{pmatrix}$$



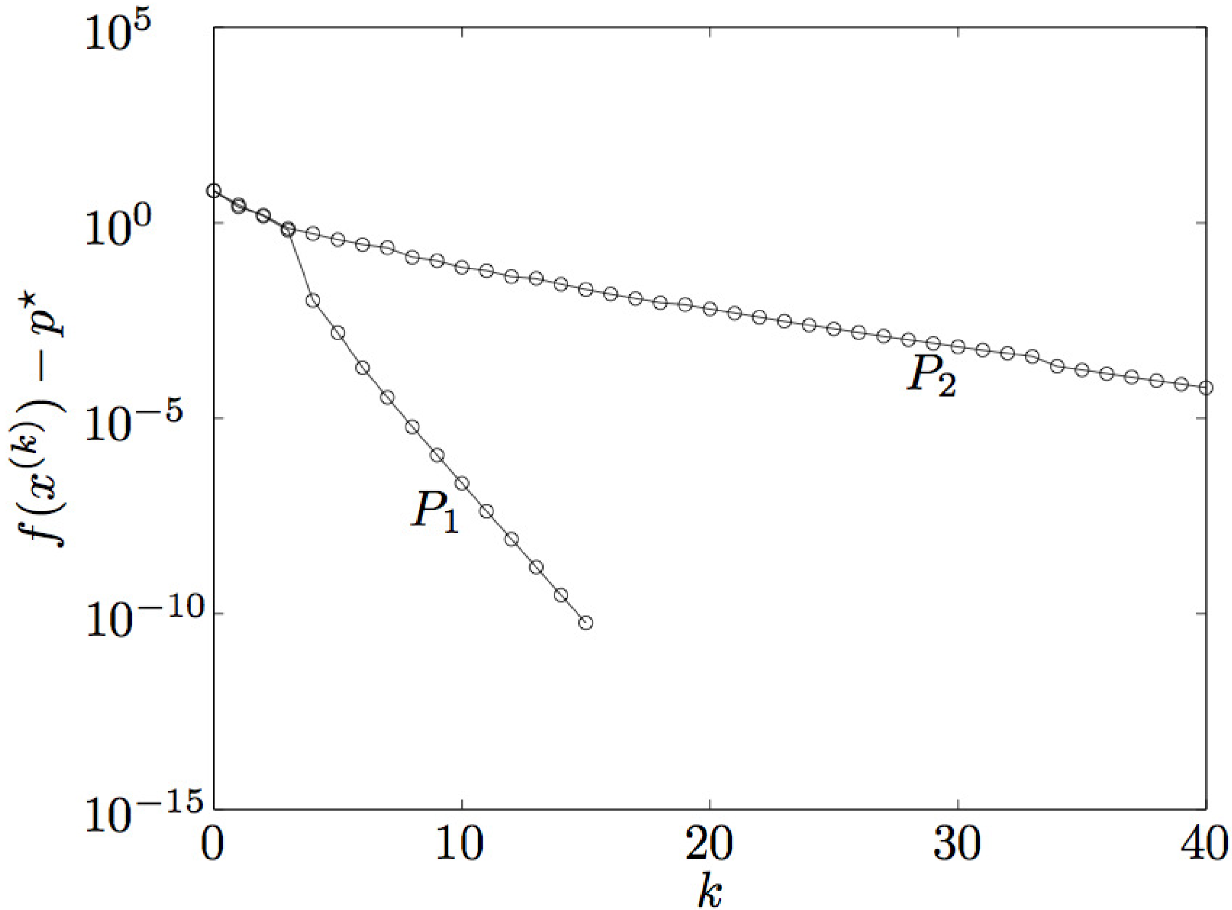



# Example: good choice

$$P = \begin{pmatrix} 2 & 0 \\ 0 & 8 \end{pmatrix}$$



# Example: comparison



# Newton's method

# The Newton step

- The vector

$$\Delta x_{nt} = -\nabla^2 f(x)^{-1} \nabla f(x)$$

  is called the *Newton step*.

- Is is a *descent direction* when the Hessian is positive semidefinite, because if $\nabla f(x) \neq 0$:

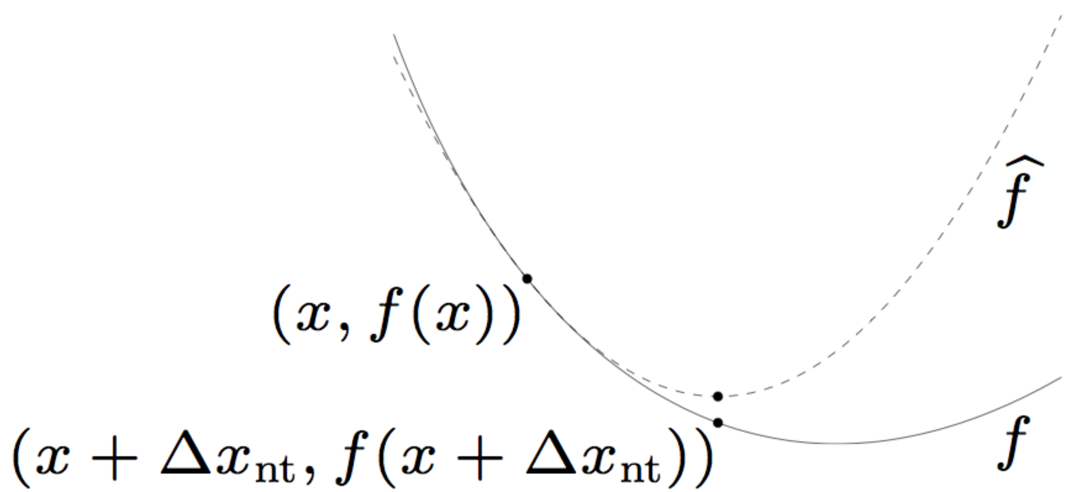
$$\nabla f(x)^\top \Delta x_{nt} = -\nabla f(x)^\top \nabla^2 f(x)^{-1} \nabla f(x) < 0$$

# Interpretation 1

- *$x + \Delta x_{nt}$ minimizes the second-order Taylor approximation of $f$ at $x$*:

$$\hat{f}(x+u) = f(x) + \nabla f(x)^\top u + \frac{1}{2} v^\top \nabla^2 f(x) v \,.$$

$\implies$ if $f$ is nearly quadratic (e.g., near its minimum for a twice differentiable function), the point $x + \Delta x_{nt}$ should be a good estimate of the minimizer $x^*$.

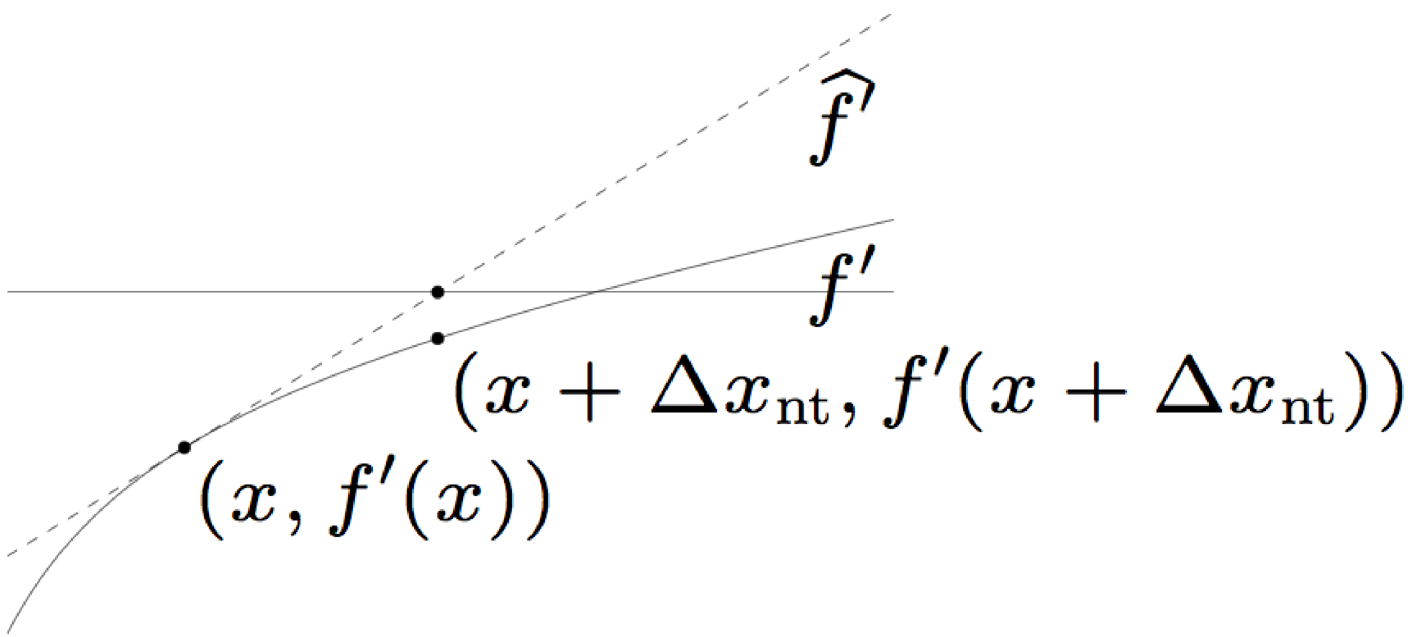


# Interpretation 2

- $x + \Delta x_{nt}$ *solves the linearized optimality condition* $\nabla f(x^*) = 0$:

$$\nabla f(x+u) \sim \nabla f(x) + \nabla^2 f(x) v = 0 \ .$$

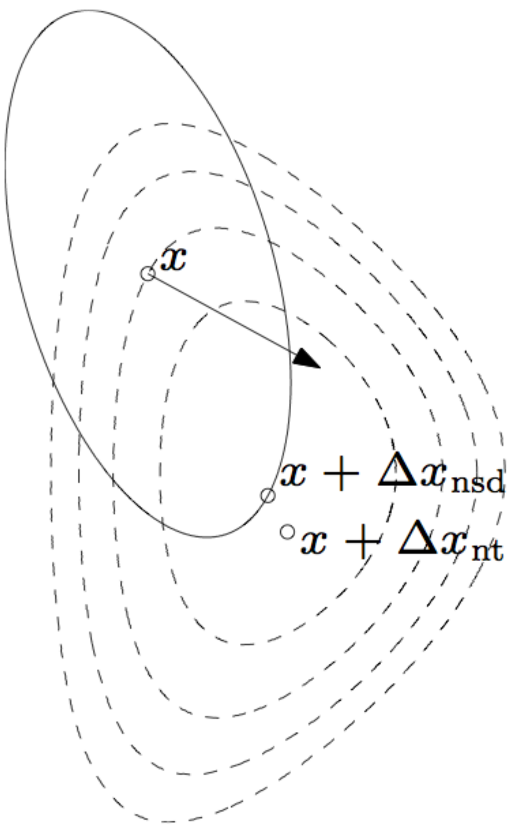
$\Longrightarrow$ this suggest again that the Newton step should be a good estimate of $x^*$ when we are already close to $x^*$.



# Interpretation 3

- *$\Delta x_{nt}$ is the steepest descent direction in the local Hessian norm:*

$$\| u \|_{\nabla^2 f(x)} = \left( u^\top \nabla^2 f(x) u \right)^{\frac{1}{2}}$$




$\Longrightarrow$ suggests fast convergence, in particular when $\nabla^2 f(x)$ is close to $\nabla^2 f(x^*)$.

# Newton decrement

The quantity

$$\lambda(x) = \left(\nabla f(x)^\top \nabla^2 f(x)^{-1} \nabla f(x)\right)^{\frac{1}{2}} ,$$

is called the *Newton decrement*, measures the proximity of $x$ to $x^*$. Several interpretations:

- gives an estimate of $f(x) - f^*$, using quadratic approximation $\hat{f}$:

$$f(x) - \inf_y \hat{f}(y) = \frac{\lambda(x)^2}{2} .$$

# Newton decrement (cont.)

- equal to the norm of the Newton step in the quadratic Hessian norm:

$$\lambda(x) = \left(\Delta x_{nt} \nabla^2 f(x) \Delta x_{nt}\right)^{\frac{1}{2}} \ .$$

- directional derivative in the Newton direction:

$$\nabla f(x)^\top \Delta x_{nt} = -\lambda(x)^2 \ .$$

- affine invariant (unlike $\| \nabla f(x) \|_2$).

# Newton's method

- *Given* a starting point $x$ and a tolerance $\epsilon > 0$.
- Repeat:
  1. Compute the Newton step and decrement:

$$\begin{cases} \Delta x_{nt} & = -\nabla^2 f(x)^{-1} \nabla f(x) \ , \\ \lambda^2 & = \nabla f(x)^\top \nabla^2 f(x)^{-1} \nabla f(x) \ . \end{cases}$$

  2. Stopping criterion: *quit* if $\lambda^2/2 \leq \epsilon$
  3. Line search: Choose step size $t$ by backtracking line search.
  4. Update: $x = x + t\Delta x_{nt}$.

*Remark: This algorithm is sometimes called the* damped *Newton method or* guarded *Newton method, to distinguish it from the* pure *Newton method which uses a fixed step size $t = 1$.*

# Convergence analysis

Suppose that:

- $f$ is *strongly convex* with constant $m$;
- $\nabla^2 f$ is Lipschitz continuous, with constant $L > 0$:

$$\| \nabla^2 f(x) - \nabla^2 f(y) \|_2 \leq L \| x - y \|_2 .$$

Then the convergence analysis is divided into two phases of the algorithm: we can show that there exists $\lambda > 0$ with:

1. the *damped Newton phase* for $\| \nabla f(x) \|_2 \geq \eta$ (slow but short)
2. the *quadratically convergent phase* for $\| \nabla f(x) \|_2 < \eta$ (fast)

# The damped Newton phase

There exists $\gamma > 0$ such that, if $\| \nabla f \left( x^{(k)} \right) \|_2 \geq \eta$, then

$$f\left(x^{(k+1)}\right) - f\left(x^{(k)}\right) \leq -\gamma$$

- Most iterations require *backtracking* steps
- The function value decreases by at least $\gamma$
- If $f^* > -\infty$, this phase ends after at most $\left( f\left(x^{(0)}\right) - f^* \right) / \gamma$ iterations.

# Quadratically convergent phase

If $\| \nabla f \left( x^{(k)} \right) \|_2 < \eta$, then

$$\frac{L}{2m^2} \| \nabla f \left( x^{(k+1)} \right) \|_2 \leq \left( \frac{L}{2m^2} \| \nabla f \left( x^{(k)} \right) \|_2 \right)^2 .$$

- All iterations use step size $t = 1$ (pure Newton)
- $\| \nabla f \left( x^{(k)} \right) \|_2$ converges to zero *quadratically*: if $\| \nabla f \left( x^{(k)} \right) \|_2 < \eta$, then for $l \geq k$:

$$\frac{L}{2m^2} \| \nabla f \left( x^{(l)} \right) \|_2 \leq \left( \frac{L}{2m^2} \| \nabla f \left( x^{(k)} \right) \|_2 \right)^{2^{l-k}} \leq \left( \frac{1}{2} \right)^{2^{l-k}} .$$

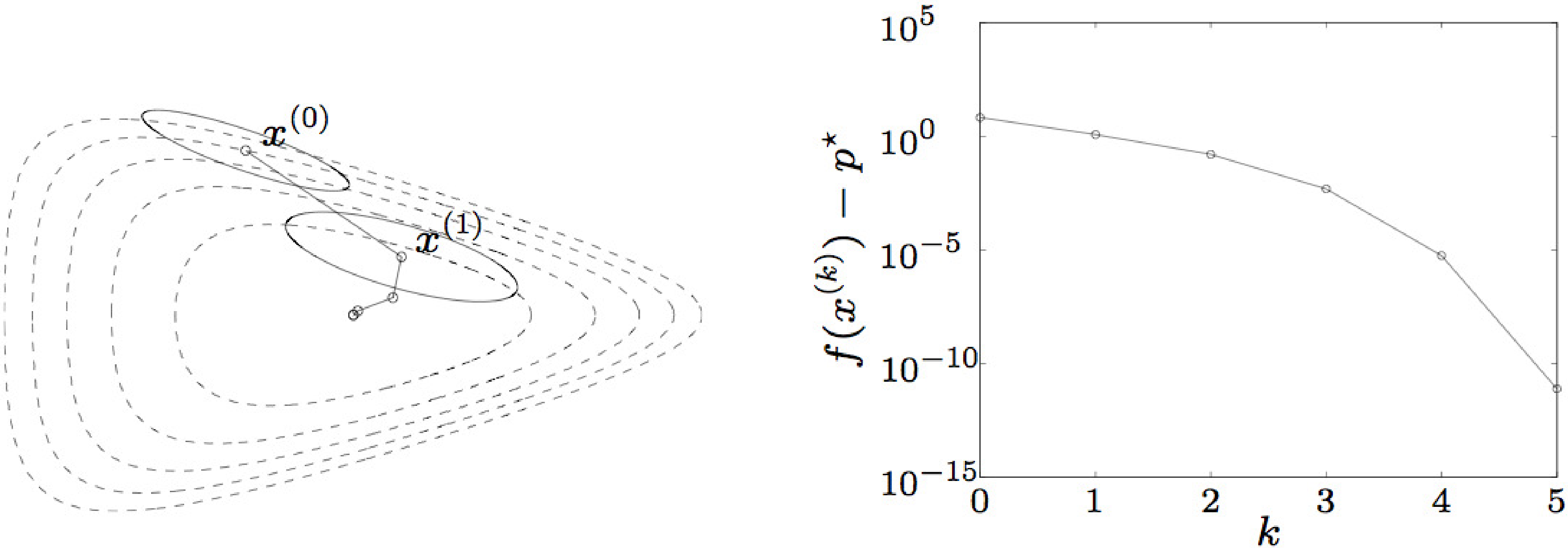
# Convergence summary

Combining the results for the two phases we see that the number of iterations until $f(x) - f^* \leq \epsilon$ is bounded above by:

$$\frac{f\left(x^{(0)}\right) - f^*}{\gamma} + \log_2 \log_2 \frac{\epsilon_0}{\epsilon} \, .$$

- $\gamma, \epsilon_0$ are constant that depend on $m, L, x^{(0)}$.
- The second term is small (*of the order of* $6$) and almost constant for practical purposes.
- In practice, constants $m, L$ (hence $\gamma, \epsilon_0$) are usually *unknown*
- This analysis provides qualitative insight in convergence properties, i.e., explains two algorithm phases.

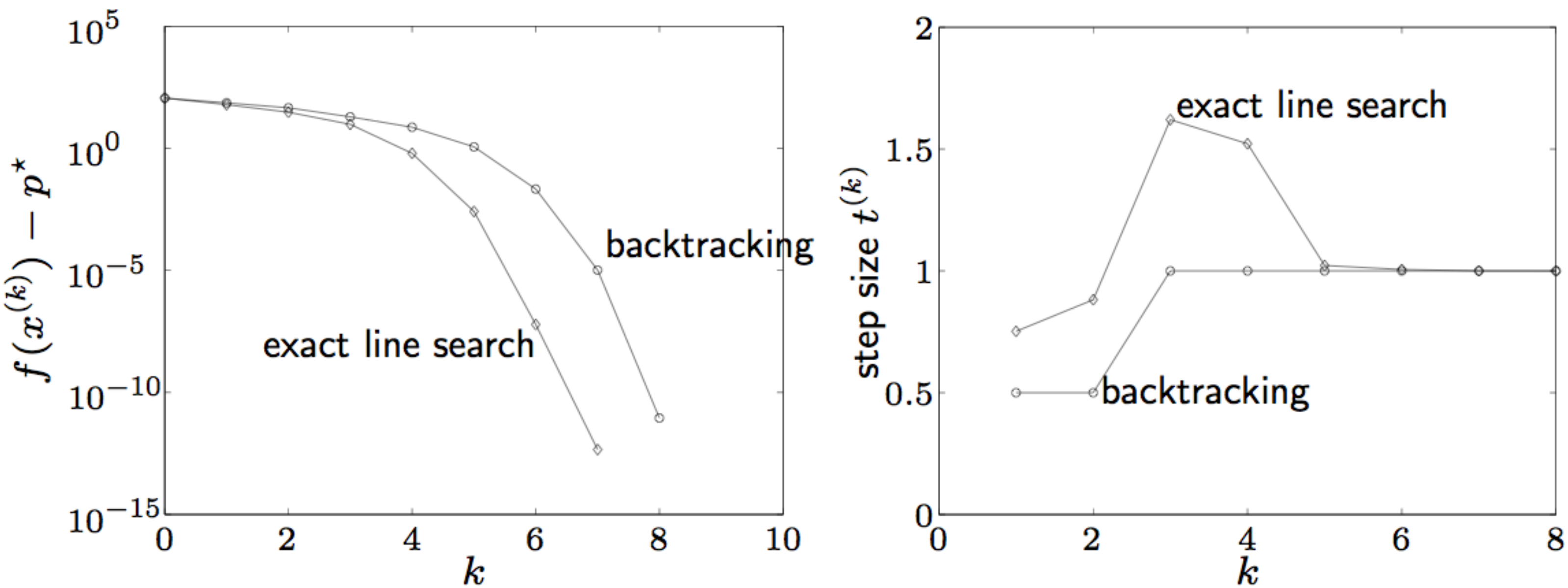
# Example




- Backtracking parameters $\alpha = 0.1$ and $\beta = 0.7$
- Converges in only $5$ iterations
- Quadratic local convergence

# Example in $\mathbb{R}^{100}$

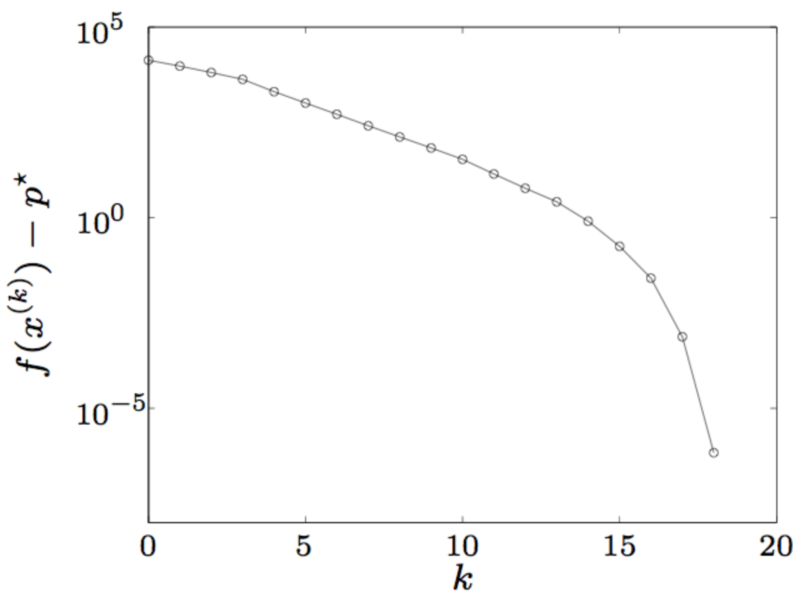
$$f(x) = c^\top x + \sum_{i=1}^{500} \log\left(b_i - a_i^\top x\right)$$




- Backtracking parameters $\alpha = 0.01$ and $\beta = 0.5$
- Backtracking line search almost as fast as exact l.s. (and much simpler)
- Clearly shows two phases in the algorithm

# Example in $\mathbb{R}^{10000}$

$$f(x) = -\sum_{i=1}^{10000} \log\left(1 - x_i^2\right) \log \sum_{i=1}^{100000} \log\left(b_i - a_i^\top x\right)$$




- Backtracking parameters $\alpha = 0.01$ and $\beta = 0.5$
- Performance similar as for small examples

# Newton's method summary

*Newton's method has several very strong advantages over gradient and steepest descent methods*:

- *Fast convergence* (at most $6$ iterations in the quadratic phase)
- *Affine invariance*: insensitive to the choice of coordinates
- *Scales well with problem size* (only a few more steps are necessary between $R^{100}$ and $R^{10000}$).
- The performance is *not dependent* on the choice of the algorithm parameters.

The main disadvantage is the *cost of forming and storing the Hessian*, and the cost of computing the Newton step.

# Implementation

Computing the Newton step $\Delta x_{nt}$ involves:

- evaluate and form the Hessian $H = \nabla^2 f(x)$ and the gradient $g = \nabla f(x)$,
- solve the linear system $H \Delta x_{nt} = -g$ (the *Newton system*, aka *normal equations*).

While general linear equation solvers can be used, it is better to use methods that take advantage of the symmetry, positive definiteness and other structures of $H$ (sparsity...).
A common approach is to use the *Cholevski* factorization $H = LL^\top$ where $L$ is lower triangular. We then solve $Lw = -g$ by forward substitution to obtain $w = -L^{-1}g$, and then solve $L^\top \Delta x_{nt} = w$ by back substitution to obtain:

$$\Delta x_{nt} = L^{-\top} w = -L^{-\top} L^{-1} g = -H^{-1} g.$$

# Conjugate gradient method

# Motivations

- Accelerate the convergence rate of steepest descent
- Avoid the overhead associated with Newton's method
- Originally developed for solving the quadratic problem:

$$\text{Minimize} \quad f(x) = \frac{1}{2} x^\top Q x - b^\top x \,,$$

  where $Q \succeq 0$, or equivalently for solving the linear system $Qx = b$.
- Generalized to non-quadratic functions

# Conjugate directions

- A set of directions $d_1, \ldots, d_k$ are *$Q$-conjugate* if:

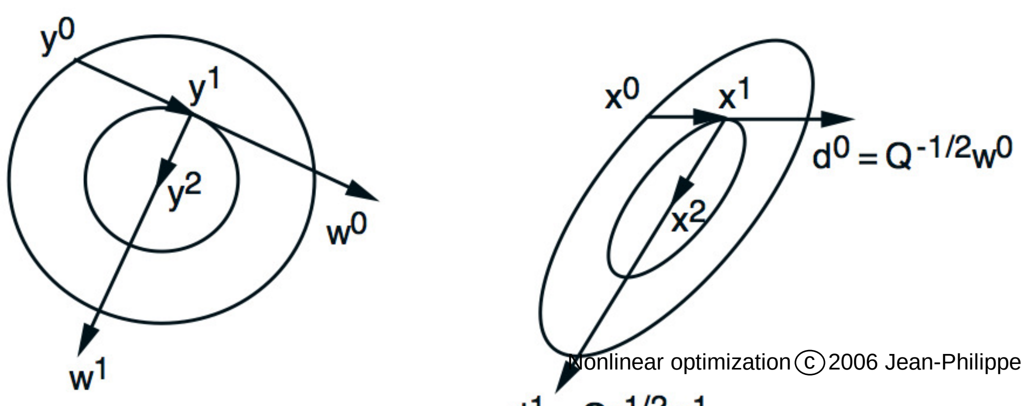
$$d_i Q d_j = 0 \quad \textit{for } i \neq j \; .$$

- If $Q$ is the identity, this is pairwise orthogonality; in general it is pairwise orthogonality of the $Q^{\frac{1}{2}} d_i$.
- Given a set of conjugated directions $d_1, \ldots, d_k$ and a new vector $\xi_{k+1}$, a conjugate direction $d_{k+1}$ is obtained by the Gram-Schmidt procedure:

$$d_{k+1} = \xi_{k+1} - \sum_{i=1}^{k} \frac{\xi_{k+1}^\top Q d_i}{d_i^\top Q d_i} d_i \; .$$

# Minimization over conjugate directions

- Let $f(x) = x^\top Q x - b^\top x$ to be minimized, $d^{(0)}, \ldots, d^{(n-1)}$ a set of $Q$-conjugate direction, $x^{(0)}$ an arbitraty starting point
- Let $x^{(k+1)} = x^{(k)} + \alpha^{(k)} d^{(k)}$ where $\alpha$ is obtained by exact line search
- Then in fact $x^{(k)}$ minimizes $f$ over the linear space spanned by $d^{(0)}, \ldots, d^{(k-1)}$: *successive iterates minimize $f$ over a progressively expanding linear manifold that eventually includes the global minimum of $f$*!



# Conjugate gradient method

- Generate conjugate directions from the successive gradients:

$$d^{(k)} = g^{(k)} - \sum_{i=1}^{k-1} \frac{g^{(i)\top} Q d^{(i)}}{d^{(i)\top} Q d^{(i)}} d^{(i)}$$

  and minimize over them.
- Key fact: the direction formula can be simplified:

$$d^{(k)} = g^{(k)} - \frac{g^{(k)\top} g^{(k)}}{g^{(k-1)\top} g^{(k-1)}} d^{(k-1)} \; .$$

- Terminates with an optimal solution with at most $n$ steps.

# Extension to non-quadratic functions

- General function $f(x)$ to be minimized
- Follow the rule $x^{(k+1)} = x^{(k)} + \alpha^{(k)} d^{(k)}$ where $\alpha^{(k)}$ is obtained by line minimization and the direction is:

$$d^{(k)} = -\nabla f\left(x^{(k)}\right) + \frac{\nabla f\left(x^{(k)}\right)^\top \left(\nabla f\left(x^{(k)}\right) - \nabla f\left(x^{(k-1)}\right)\right)}{\nabla f\left(x^{(k-1)}\right)^\top \nabla f\left(x^{(k-1)}\right)} d^{(}$$

- Due to non-quadratic function and numerical errors, conjugacy is progressively lost $\implies$ operate the method in *cycles of conjugate direction steps*, with the first step in a cycle being a steepest direction.

# Summary

- Converges in $n$ steps for a quadratic problem
- Limited memory requirements
- A good line search is required to limit the loss of direction conjugacy (and the attendant deterioration of convergence rate).

# Quasi-Newton methods

# Motivations

- Quasi-Newton methods are gradient methods of the form:
$$\begin{cases} x^{(k+1)} & = x^{(k)} + \alpha^{(k)} d^{(k)} \,, \\ d^{(k)} & = -D^{(k)} \nabla f \left( x^{(k)} \right) \,, \end{cases}$$
where $D^{(k)}$ is a p.d. matrix which may be adjusted from one iteration to the next one to approximate the inverse Hessian.
- Goal: approximate Newton's method without the burden of computing and inverting the Hessian

# Key Idea

Successive iterates $x^{(k)}, x^{(k+1)}$ and gradients $\nabla f\left(x^{(k)}\right), \nabla f\left(x^{(k+1)}\right)$ yield *curvature information*:

$$q_k \sim \nabla^2 f\left(x^{k+1}\right) p_k \ ,$$

with

$$\begin{cases} p_k & = x^{(k+1)} - x^{(k)} \ , \\ q_k & = \nabla f\left(x^{(k+1)}\right) - \nabla f\left(x^{(k)}\right) \ . \end{cases}$$

This idea has been translated into several quasi-Nexton algorithms

# Davidon-Fletcher-Powell (DFP) method

- The first and best-known quasi-gradient method
- The successive inverse Hessian approximations are constructed by the formula:

$$D^{(k+1)} = D^{(k)} + \frac{p_k p_k^\top}{p_k^\top q_k} - \frac{D^{(k)} q_k q_k^\top D^{(k)}}{q_k^\top D^{(k)} q_k}$$

# Summary

- Typically converges fast
- Avoid the explicit second derivative calculations of Newton's method
- Main drawback relative to the conjugate gradient method:
  - requires the storage of the approximated Hessian
  - requires a matrix-vector multiplication to compute the direction

# Conclusion

# Summary

- Do not use simple gradient descent
- If you can afford it (in time and memory), use Newton's method.
- For non-convex problem, be careful in the first iterations
- If inverting the Hessian is not possible, quasi-Newton is a good alternative.
- Conjugate gradient requires no matrix storage, but should be done more carefully (loss of conjugacy).