

**This lecture:**

- Gradient descent methods
- Choosing the descent direction
- Choosing the step size
- Convergence
- Convergence rate

Let's recall our unconstrained optimization problem:

$$\min_x f(x), \quad (f: \mathbb{R}^n \rightarrow \mathbb{R})$$

**Where we stand so far:**

- Learned about some structural properties of local optimal solutions (first and second order conditions for optimality).
- Learned that for convex problems, local optima are automatically global.
- But how to find a local optimum?
- How to even find a stationary point (i.e., a point where the gradient vanishes)? Recall that this would suffice for global optimality if  $f$  is convex.
- We now begin to see some algorithms for this purpose, starting with gradient descent algorithms.
- These will be iterative algorithms: start at a point, jump to a new point that hopefully has a lower objective value and continue.
- Our presentation uses references [Bert03], [Bert09], [Tit13], [CZ13].

**General form of the iterations:**

$$x_{k+1} = x_k + \alpha_k d_k$$

$k \in \mathbb{Z}_+$ : index of time (iteration number)

$x_k \in \mathbb{R}^n$ : Current point       $d_k \in \mathbb{R}^n$ : Direction to move along at iteration  $k$

$x_{k+1} \in \mathbb{R}^n$ : Next point       $\alpha_k \in \mathbb{R}_+$ : Step size at iteration  $k$

Goal is to make the sequence  $\{f(x_k)\}$  decrease as much as possible.

- How to choose  $d_k$ ? How to choose  $\alpha_k$ ?

## Gradient methods

- Throughout this lecture, we assume that  $f$  is at least  $C^1$  (continuously differentiable).
- Gradient methods: the direction  $d_k$  to move along at step  $k$  is chosen "based on information from"  $\nabla f(x_k)$ .
- Why is  $\nabla f(x_k)$  a natural vector to look at? Lemmas 1 and 2 below (proved on the next page) provide two reasons.

**Lemma 1:** Consider yourself sitting at a point  $x \in \mathbb{R}^n$  and looking (locally) at the value of the function  $f$  in all directions around you. The direction with the maximum rate of decrease is along  $-\nabla f(x)$ .

**Remark:** When we speak of direction, the magnitude of the vector does not matter; e.g.,  $\nabla f(x)$ ,  $5\nabla f(x)$ ,  $\frac{\nabla f(x)}{20}$ ,  $\frac{\nabla f(x)}{\|\nabla f(x)\|}$  all are in the same direction.

**Definition:** For a given point  $x \in \mathbb{R}^n$ , a direction  $d \in \mathbb{R}^n$  is called a **descent direction**, if there exists  $\bar{\alpha} > 0$  ( $\alpha \in \mathbb{R}$ ) such that

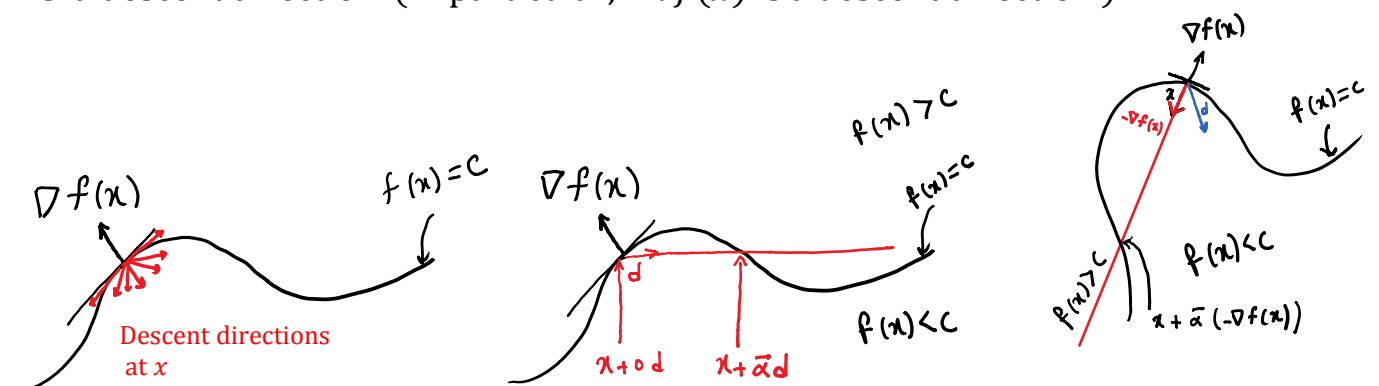
$$f(x + \alpha d) < f(x), \quad \forall \alpha \in (0, \bar{\alpha}).$$

**Interpretation:** There is a small enough (but nonzero) amount that you can move in direction  $d$  and be guaranteed to decrease the function value.

**Lemma 2:** Consider a point  $x \in \mathbb{R}^n$ . Any direction  $d$  satisfying

$$\langle d, \nabla f(x) \rangle < 0$$

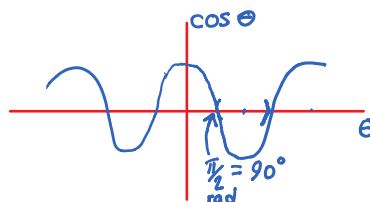
is a descent direction. (In particular,  $-\nabla f(x)$  is a descent direction.)



- Red: direction of steepest descent
- Blue: Descent direction  $d$  would have been better

**Remark:** The condition  $\langle d, \nabla f(x) \rangle < 0$  in Lemma 2 geometrically means that the vectors  $d$  and  $\nabla f(x)$  make an angle of more than 90 degrees (on the plane that contains them).

Why?  $\langle d, \nabla f(x) \rangle = \|d\| \cdot \|\nabla f(x)\| \cos(\theta)$



**Proof of Lemma 1.**

Consider a point  $x$ , a direction  $d$ , and the univariate function

$$g(\alpha) = f\left(x + \alpha \frac{d}{\|d\|}\right).$$

The rate of change of  $f$  at  $x$  in direction  $d$  is given by  $g'(0)$ , which by the chain rule equals

$$\frac{1}{\|d\|} \langle \nabla f(x), d \rangle.$$

By the Cauchy-Schwarz inequality (see, e.g., Theorem 2.3 of [CZ13] for a proof), we have:

$$-\frac{1}{\|d\|} \cdot \|\nabla f(x)\| \cdot \|d\| \leq \frac{1}{\|d\|} \langle \nabla f(x), d \rangle \leq \frac{1}{\|d\|} \cdot \|\nabla f(x)\| \cdot \|d\|,$$

which after simplifying gives

$$-\|\nabla f(x)\| \leq \frac{1}{\|d\|} \langle \nabla f(x), d \rangle \leq \|\nabla f(x)\|.$$

So the rate of change in any direction cannot be larger than  $\|\nabla f(x)\|$ , or smaller than  $-\|\nabla f(x)\|$ . However, if we take  $d = \nabla f(x)$ , the right inequality is achieved:

$$\frac{1}{\|\nabla f(x)\|} \langle \nabla f(x), \nabla f(x) \rangle = \frac{1}{\|\nabla f(x)\|} \cdot \|\nabla f(x)\|^2 = \|\nabla f(x)\|.$$

Similarly, if we take  $d = -\nabla f(x)$ , then the left inequality is achieved.

**Proof of Lemma 2.**

By Taylor's theorem, we have

$$f(x + \alpha d) = f(x) + \alpha \nabla f(x)^T d + o(\alpha).$$

Since  $\lim_{\alpha \rightarrow 0} \frac{|o(\alpha)|}{\alpha} = 0$ , there exists  $\bar{\alpha} > 0$  such that

$$\frac{|o(\alpha)|}{\alpha} < |\nabla f(x)^T d|, \quad \forall \alpha \in (0, \bar{\alpha}).$$

This, together with our assumption that  $\nabla f(x)^T d < 0$ , implies that  $\forall \alpha \in (0, \bar{\alpha})$  we must have:

$$f(x + \alpha d) - f(x) < 0.$$

Hence,  $d$  is a descent direction.

**Lemma 3:** Consider any positive definite matrix  $B$ . For any point  $x$ , with  $\nabla f(x) \neq 0$ , the direction  $-B\nabla f(x)$  is a descent direction.

**Proof.** We have  $\langle -B\nabla f(x), \nabla f(x) \rangle = -\nabla f(x)^T B \nabla f(x) < 0$ , by the assumption that  $B$  is positive definite.

This suggests a general paradigm for our descent algorithms:

$$x_{k+1} = x_k - \alpha_k B_k \nabla f(x_k), \quad (\text{with } B_k \succ 0, \forall k).$$

### Common choices of descent direction:

- **Steepest Descent:**  $B_k = I, \forall k$ .

Simplest descent direction but not always the fastest.

- **Newton Direction:**  $B_k = (\nabla^2 f(x_k))^{-1}$  (assuming Hessian positive definite).

More expensive, but can have much faster convergence.

- **Diagonally Scaled Steepest Descent:**  $B_k = \text{diag}(d_{1,k}, \dots, d_{n,k}), d_{i,k} > 0$ .

For example, can take  $d_{i,k} \approx \left( \frac{\partial^2 f(x_k)}{(\partial x_i)^2} \right)^{-1}$ ; i.e., diagonally approximate Newton.

- **Modified Newton Direction:**  $B_k = (\nabla^2 f(x_0))^{-1}, \forall k$

Compute Newton direction only at the beginning, or once every  $M$  steps.

- **Quasi-Newton Directions:** Chap. 11 of [CZ13].

Tradeoffs generally problem dependent.

## Common choices of the step size $\alpha_k$ .

Back to the general form of our iterative algorithm:

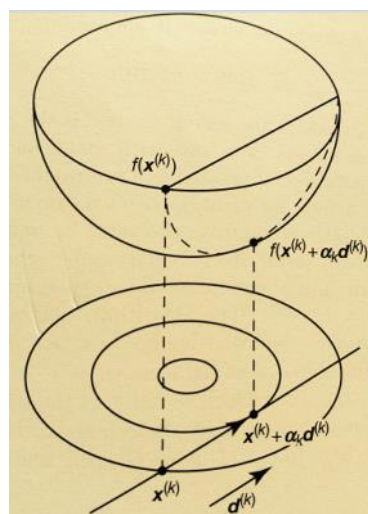
$$x_{k+1} = x_k + \alpha_k d_k$$

- **Constant step size:**  $\alpha_k = s, \forall k$  ( $s > 0$ )
  - Simplest rule to implement, but may not converge if  $s$  too large; may be too slow if  $s$  too small.
- **Diminishing step size:**  $\alpha_k \rightarrow 0, \sum_{k=1}^{\infty} \alpha_k = \infty$ . (e.g.,  $\alpha_k = \frac{1}{k}$ )
  - Descent not guaranteed at each step; only later when  $\alpha_k$  becomes small.
  - $\sum_{k=1}^{\infty} \alpha_k = \infty$  imposed to guarantee progress does not become too slow.
  - Good theoretical guarantees, but unless the right sequence is chosen, can also be a slow method.
- **Minimization rule (exact line search):**  $\alpha_k = \operatorname{argmin}_{\alpha \geq 0} f(x_k + \alpha d_k)$ 
  - A minimization problem itself, but an easier one (one dimensional).
  - Can use the line search methods that we learned in the previous lecture.
  - No need to be very precise at each step.
  - If  $f$  convex, the one dimensional minimization problem also convex (why?).
- **Limited minimization:**  $\alpha_k = \operatorname{argmin}_{\alpha \in [0, s]} f(x_k + \alpha d_k)$ 
  - Previous comments apply.
  - Tries not to step too far.
- **Successive step size reduction:** well-known examples are Armijo rule and Goldstein rule (see, e.g., Section 7.8 of [CZ13]). We will cover Armijo in the next lecture.
  - Try to ensure enough decrease in line search without spending time to solve it to optimality.

Tradeoffs generally problem dependent.

An illustration of a single step of the minimization rule (aka exact line search) for choosing the step size.

Image credit: [CZ13]



## Stopping criteria.

- Once we have a rule for choosing the search direction and the step size, we are good to go for running the algorithm.
- Typically the initial point  $x_0$  is picked randomly, or if we have a guess for the location of local minima, we pick  $x_0$  close to them.
- But when to stop the algorithm?
- Some common choices ( $\epsilon > 0$  is a small prescribed threshold):
  - $\|\nabla f(x)\| < \epsilon$ 
    - Note: if we have  $\nabla f(x_k) = 0$ , our iterates stop moving. We have found a point satisfying the first order necessary condition for optimality. This is what we are aiming for.
  - $|f(x_{k+1}) - f(x_k)| < \epsilon$ 
    - Improvements in function value are saturating.
  - $\|x_{k+1} - x_k\| < \epsilon$ 
    - Movement between iterates has become small.
  - $\frac{|f(x_{k+1}) - f(x_k)|}{\max\{1, |f(x_k)|\}} < \epsilon$ 
    - A "relative" measure - removes dependence on the scale of  $f$ .
    - The max is taken to avoid dividing by small numbers.
  - $\frac{\|x_{k+1} - x_k\|}{\max\{1, \|x_k\|\}} < \epsilon$ 
    - Same comments apply.

## Steepest descent with exact line search for quadratic functions.

Goal:  $\min_x f(x) = \frac{1}{2} x^T Q x - b^T x$ , assume  $Q \succ 0$ .

- o If  $Q$  has a negative eigenvalue,  $f^* = -\infty$  (why?)
- o If  $Q \succ 0 \Rightarrow$  unique global solution (why?)

$$x_{k+1} = x_k - \alpha_k \nabla f(x_k)$$

Goal: Choose  $\alpha_k \geq 0$  to minimize  $f(x_{k+1})$ .

$$\text{Let } g(\alpha) := f(x_k - \alpha \nabla f(x_k))$$

$$= \frac{1}{2} (x_k - \alpha \nabla f(x_k))^T Q (x_k - \alpha \nabla f(x_k)) - b^T (x_k - \alpha \nabla f(x_k))$$

- o  $g(\alpha)$  is quadratic and convex (why?)

$$g(\alpha) = a \alpha^2 + d \alpha + c \rightarrow \arg \min g(\alpha) = -\frac{d}{2a}$$

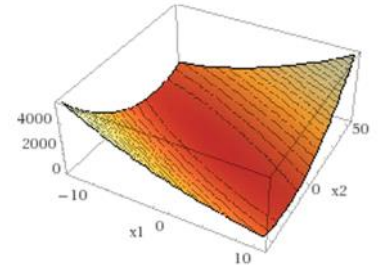
$$\text{where } a = \frac{1}{2} \nabla f^T(x_k) Q \nabla f(x_k), d = (b^T - x_k^T Q) \nabla f(x_k) = -\nabla f^T(x_k) \nabla f(x_k)$$

$$\Rightarrow \alpha_k = \frac{\nabla f^T(x_k) \nabla f(x_k)}{\nabla f^T(x_k) Q \nabla f(x_k)}$$

Overall Algorithm:

$$x_{k+1} = x_k - \left( \frac{\nabla f^T(x_k) \nabla f(x_k)}{\nabla f^T(x_k) Q \nabla f(x_k)} \right) \nabla f(x_k)$$

$$\text{where } \nabla f(x_k) = Q x_k - b$$



**Example.**

$$\min. f(x) = 5x_1^2 + x_2^2 + 4x_1x_2 - 6x_1 - 4x_2 + 15$$

Any stationary point must be the unique global minimizer (why?)

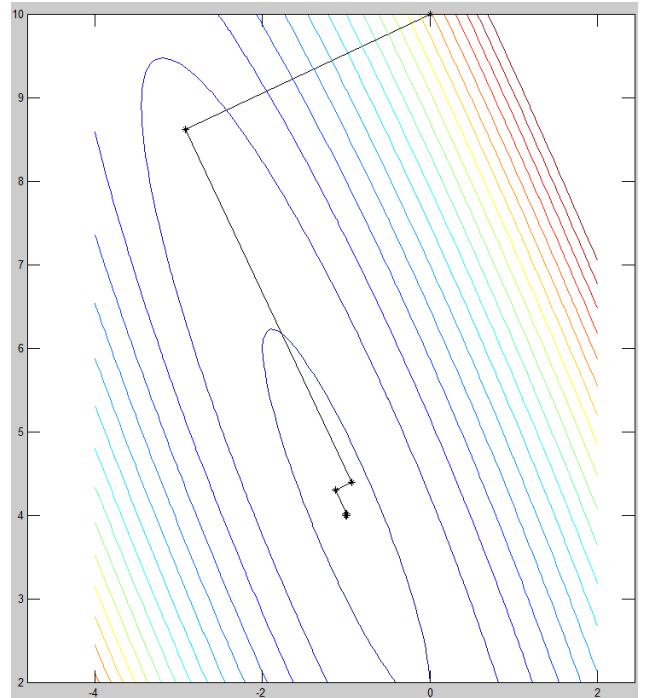
Let's try the steepest descent method:

$$d_k = -\nabla f(x_k)$$

$\alpha_k$ : get from exact line search

$$x_0 = (0, 10)^T$$

Stopping criterion:  $\|\nabla f(x)\| < 10^{-6}$



**Output:**

$K$	$x_k$		$\nabla f(x_k)$		$\ \nabla f(x_k)\ $	$\alpha_k$	$f(x_k)$
1.	0.0000000000000000	0	10.0000000000000000	34.0000000000000000	16.0000000000000000	37.5765888456111871	75.0000000000000000
2.	-2.923039454456893	8.624452021432051	-0.732586458840725	1.556746225036530	1.720506242023632	2.715384615384604	14.303945445689237
3.	-0.933785454681702	4.397287271909798	2.251294540822171	1.059432725092788	2.488116719234297	0.085971748660497	10.284983790761091
4.	-1.127333183106014	4.306205987945416	-0.048507879278480	0.1030792434466774	0.113922538532891	2.715384615384754	10.018870072128331
5.	-0.995615633988288	4.026306196070238	0.149068444398068	0.070149856187323	0.164749517262910	0.085971748660497	10.001249473246101
6.	-1.008431308823290	4.020275290265531	-0.003211927170778	0.006825345237901	0.007543329090456	2.715384615384132	10.000082733302879
7.	-0.9997909691198026	4.001741852811849	0.009870499267134	0.004644940831593	0.010908814376984	0.085971748660486	10.000005478148033
8.	-1.000558275280174	4.001342519126133	-0.000212676297206	0.000451937131571	0.000499478105912	2.715384615432194	10.00000362733084
9.	-0.999980777334673	4.000115335991923	0.000653570620962	0.000307562645155	0.000722322183848	0.085971748660553	10.00000024018206
10.	-1.000036965943830	4.000088894293497	-0.000014082264316	0.000029924811672	0.000033072715672	2.715384615356879	10.00000001590355
11.	-0.99998727179956	4.000007636920265	0.000043275881499	0.000020365120705	0.000047828234975	0.085971748665113	10.00000000105302
12.	-1.000002447683163	4.000005886095226	-0.000000932450728	0.000001981457800	0.000002189894831	-0.692156863782812	10.000000000006972
13.	-1.000003093085535	4.000007257574842	-0.000001900553979	0.000002142808345	0.000002864215954	0.644738427673033	10.00000000010715
14.	-1.000001867725151	4.000005876023959	0.000004826844329	0.000004281147315	0.000006451871706	0.095703090047471	10.000000000008070
15.	-1.000002329669068	4.000005466304932	-0.000001431470952	0.000001613933592	0.000002157287817	0.644738430768048	10.000000000006079
16.	-1.000001406744733	4.000004425739920	0.000003635512357	0.000003224500911	0.000004859460487	0.095703089988433	10.000000000004576
17.	-1.000001754674499	4.000004117145219	-0.000001078164111	0.000001215592444	0.000001624839327	0.644738428822084	10.000000000003446
18.	-1.000001059540664	4.000003333406057	0.000002738217589	0.000004286494458	0.000003660078381	0.095703090024235	10.000000000002597
19.	-1.000001321596548	4.0000033100976799	-0.000000812058286	0.000000915567405	0.000001223806493	-0.251801315258460	10.000000000001959
20.	-1.000001526073893	4.000003331517876	-0.000001934667424	0.000000558740181	0.000002013734995	0.140893244247662	10.000000000002405
21.	-1.000001253492323	4.000003252795159	0.000000476257405	0.000001491621026	0.000001565807907	0.241956665294677	10.000000000002125
22.	-1.000001368725976	4.000002891887509	-0.00000219709727	0.000000308871113	0.000002142094930	0.115598723360049	10.000000000001897
23.	-1.000001123690238	4.000002856182403	0.000000187827231	0.000001217603853	0.000001232005768	0.511898030938667	10.000000000001631
24.	-1.000001219838628	4.000002232893388	-0.000003266812726	-0.000000413567736	0.000003292886828	0.092258931075849	10.000000000001528
25.	-1.000000918445978	4.000002271048705	-0.000000100264955	0.000000868313499	0.000000874083174	0.000000001000000	10.000000000001036



## Lec8p9, ORF363/COS323

**Theorem.** ("the zig-zag theorem") Let  $\{x_k\}$  be the sequence generated by the steepest descent algorithm. Then, for all  $k$ ,  $x_{k+1} - x_k$  is orthogonal to  $x_{k+2} - x_{k+1}$ .

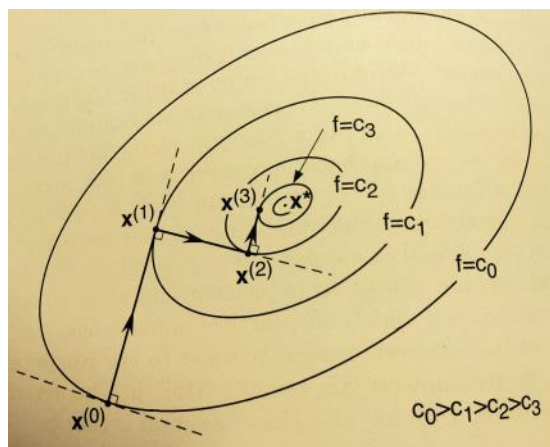


Image credit: [CZ13]

**Proof.** Our iterations read:

$$x_{k+1} = x_k - \alpha_k \nabla f(x_k)$$

$$x_{k+2} = x_{k+1} - \alpha_{k+1} \nabla f(x_{k+1}).$$

Hence,

$$\langle x_{k+1} - x_k, x_{k+2} - x_{k+1} \rangle = \alpha_k \alpha_{k+1} \langle \nabla f(x_k), \nabla f(x_{k+1}) \rangle. \quad (1)$$

Recall that  $\alpha_k$  minimizes  $\varphi_k(\alpha) := f(x_k - \alpha \nabla f(x_k))$ .

$$\Rightarrow \frac{d\varphi_k}{d\alpha}(\alpha_k) = 0. \quad (2)$$

But by chain rule:

$$\begin{aligned} \frac{d\varphi_k}{d\alpha}(\alpha_k) &= \langle -\nabla f(x_k), \nabla f(x_k - \alpha_k \nabla f(x_k)) \rangle \\ &= \langle -\nabla f(x_k), \nabla f(x_{k+1}) \rangle. \quad (3) \end{aligned}$$

$$(1) + (2) + (3) \Rightarrow \langle x_{k+1} - x_k, x_{k+2} - x_{k+1} \rangle = 0. \quad \square$$

## Convergence

The descent algorithms discussed so far typically come with proofs of convergence to a stationary point (i.e., a point where the gradient of the objective function is zero). We state a couple such theorems here; the proofs are a bit tedious, and I won't require you to know them. But those interested can look some of these proofs up in [CZ13].

**Theorem.** (See [Bert03].) Consider the sequence  $\{x_k\}$  generated by any descent algorithm with

$$d_k = -B_k \nabla f(x_k),$$

such that eigenvalues of  $B_k$  are larger than some  $m > 0$ , for all  $k$ , and the step size is chosen according to the minimization rule, or the limited minimization rule, (or the Armijo rule).

Then, every limit point of  $\{x_k\}$  is a stationary point.

**Remark1:** We say  $x \in \mathbb{R}^n$  is a limit point of a sequence  $\{x_k\}$ , if there exists a subsequence of  $\{x_k\}$  that converges to  $x$ .

**Remark2:** A stationary point may not be a local min! It may be a saddle point or even a local max! But there is a result called the "capture theorem" (see [Bert03]) which informally states that isolated local minima tend to attract gradient methods.

**Question to think about:** How would you check if the point that you end up with (assuming it is stationary) is actually a local minimum?

**Theorem.** (See [CZ13].) Consider a quadratic function

$$f(x) = \frac{1}{2} x^T Q x + b^T x + c, \text{ with } Q \succ 0, \text{ and let } x_* \text{ be the}$$

minimizer.

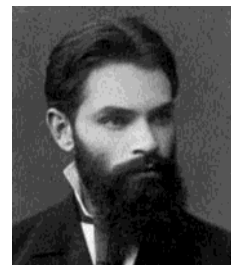
- For the steepest descent algorithm with exact line search, we have  $x_k \rightarrow x_*$  starting from any  $x_0$ . (This is called global convergence.)
- For the steepest descent algorithm with a fixed step size, we have global convergence if and only if the step size  $\alpha$  satisfies:

$$0 < \alpha < \frac{2}{\lambda_{\max}(Q)},$$

where  $\lambda_{\max}(Q)$  denotes the maximum eigenvalue of  $Q$ .

## Lyapunov functions and proofs of convergence

- A very powerful technique for proving convergence of iterative algorithms is based on the idea of a Lyapunov function.
- This is a technique from the theory of dynamical systems and control. Roughly speaking, it is used to decide whether all trajectories of a dynamical system converge to a given point.
- The idea is due to the Russian mathematician, Aleksandr Lyapunov.



[Lyapunov](#)  
(1857-1918)

- We will typically try to prove that all trajectories of a dynamical system go to the origin; i.e., we want to show that for all  $x_0 \in \mathbb{R}^n$ ,  $x_k \rightarrow 0$ , as  $k \rightarrow \infty$ .
  - In control theory, the vector  $x_k$  can denote the deviation of an airplane wing angles from its stable flying position at time  $k$ .
  - When proving convergence of iterative algorithms,  $x_k$  can denote the deviation of the  $k$ -th point that our algorithm produces from the optimal solution  $x_*$ . (A better notation is perhaps  $e_k$ ; for *error* at step  $k$ .)

**Theorem.** ("Lyapunov's global stability theorem in discrete time")

Consider a dynamical system

$$x_{k+1} = g(x_k)$$

where  $g: \mathbb{R}^n \rightarrow \mathbb{R}^n$  is a continuous map satisfying  $g(0) = 0$ .

If there exists a continuous function  $V: \mathbb{R}^n \rightarrow \mathbb{R}$  (called a "Lyapunov function")

that satisfies:

- $V(0) = 0$ ,  $V(x) > 0 \quad \forall x \neq 0$ , (positivity)
- $V(x) \rightarrow \infty$  as  $\|x\| \rightarrow \infty$ , (radial unboundedness)
- $V(g(x)) < V(x) \quad \forall x \neq 0$ , (strict decrease)

then,  $\forall x_0 \in \mathbb{R}^n$  we have  $x_k \rightarrow 0$  as  $k \rightarrow \infty$ .

**Proof.**

The decrease assumption of the theorem implies that the function  $V$  monotonically decreases along all trajectories of our dynamical system:



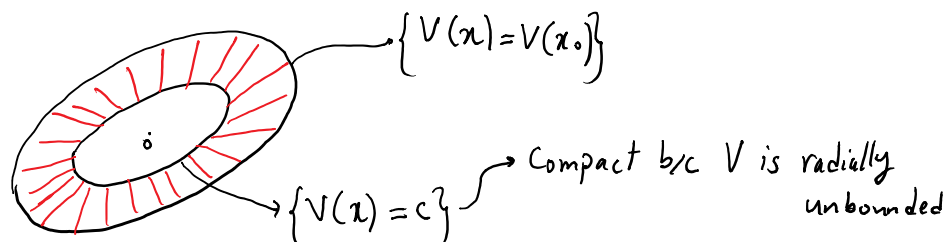
Take any  $x_0$  and consider the sequence  $\{V(x_k)\}$  of the function  $V$  evaluated on the trajectory starting at  $x_0$ .

Since  $\{V(x_k)\}$  is positive and lower bounded, it converges to some  $c \geq 0$ .

If  $c = 0$ ,  $V(x_k) \rightarrow 0$  implies that  $x_k \rightarrow 0$  (because  $V$  is only zero at zero) and we would be done.

We claim that we cannot have  $c > 0$ . Indeed if  $c > 0$ , then the trajectory starting at  $x_0$  would forever be traversing in the compact set

$$S := \{x \mid V(x) \geq c\} \cap \{x \mid V(x) \leq V(x_0)\}.$$



Let  $\epsilon := \min_{x \in S} |V(g(x)) - V(x)|$ . Since the objective is continuous and



## Rates of convergence

- Once we know an iterative algorithm converges, the next question is how fast?
- If  $|f(x_k) - f(x_*)| \approx \frac{1}{\log(\log(k))}$ , then sure, this difference will go to zero eventually. But to see it go even below 0.1, you need to write a letter to your grandson (to tell him to write a letter to his grandson, etc.)
- Let us formalize this.

**Definition** (see [Tit13]). Let  $\{x_k\}$  converge to  $x_*$ . We say the convergence is of **order**  $p (\geq 1)$  and with **factor**  $\gamma (> 0)$ , if  $\exists k_0$  such that  $\forall k \geq k_0$ ,

$$\|x_{k+1} - x_*\| \leq \gamma \|x_k - x_*\|^p.$$

Make sure the following comments make sense:

- The larger the power  $p$ , the faster the convergence.
- For the same  $p$ , the smaller  $\gamma$ , the faster the convergence.
- If  $\{x_k\}$  converges with order  $p$ , it also converges with order  $p'$  for any  $p' \leq p$ .
- If  $\{x_k\}$  converges with order  $p$  and factor  $\gamma$ , it also converges with order  $p'$  and factor  $\gamma'$ , for any  $\gamma' \geq \gamma$ .
- So we typically look for the largest  $p$  and the smallest  $\gamma$  for which the inequality holds.

### Some more terminology:

- If  $p = 1$ , and  $\gamma < 1$ , we say convergence is **linear**.
- If  $p = 1$ , and  $\gamma = 1$ , we say convergence is **sublinear**.
- If  $p > 1$ , we say that convergence is **superlinear**. (This is slightly stronger than the usual definition of superlinear convergence, but we will go with it.)
- If  $p = 2$ , we say that convergence is **quadratic**.

- Why called linear convergence?

For  $k$  large enough, we have  $\|x_{k+i} - x_*\| \leq \gamma^i \|x_k - x_*\|$ . So  $\log \|x_{k+i} - x_*\| \leq i \log \gamma + \log \|x_k - x_*\|$ . Hence,  $-\log \|x_{k+i} - x_*\|$ , which is a measure of the number of correct significant digits in  $x_{k+i}$ , grows linearly with  $i$ .

$$-\log_{10} 0.1 = 1, \quad -\log_{10} 0.01 = 2, \quad -\log_{10} 0.001 = 3, \quad \text{etc.}$$

## Examples:

- $\|x_k - x_*\| \approx \frac{1}{k}$  sublinear convergence.
- $\|x_k - x_*\| \approx a^k, 0 < a < 1$  linear convergence.
- $\|x_k - x_*\| \approx a^{2^k}, 0 < a < 1$  quadratic convergence.  
(Verify!)
- Quadratic convergence is super fast! Number of correct significant digit doubles in each iteration (why?).

## Convergence rate of steepest descent for quadratic functions

**Theorem.** Consider a quadratic function  $f(x) = \frac{1}{2}x^T Qx + b^T x + c$ , with  $Q \succ 0$ . Let  $m$  and  $M$  respectively denote the smallest and largest eigenvalue of  $Q$ . Then the sequence  $\{f(x_k)\}$  generated by the steepest descent algorithm with exact line search converges to the unique global minimum of  $f$ , where the convergence is linear (order 1), and with factor  $\left(\frac{M-m}{M+m}\right)^2$ .

- $\kappa := \frac{M}{m}$  is called the condition number of the matrix  $Q$ . (Note  $\kappa \geq 1$ .)
  - Very important quantity in numerical analysis
- Note:  $\left(\frac{M-m}{M+m}\right)^2 = \left(\frac{\kappa-1}{\kappa+1}\right)^2$
- We want  $\kappa$  small for fast convergence (close to one).

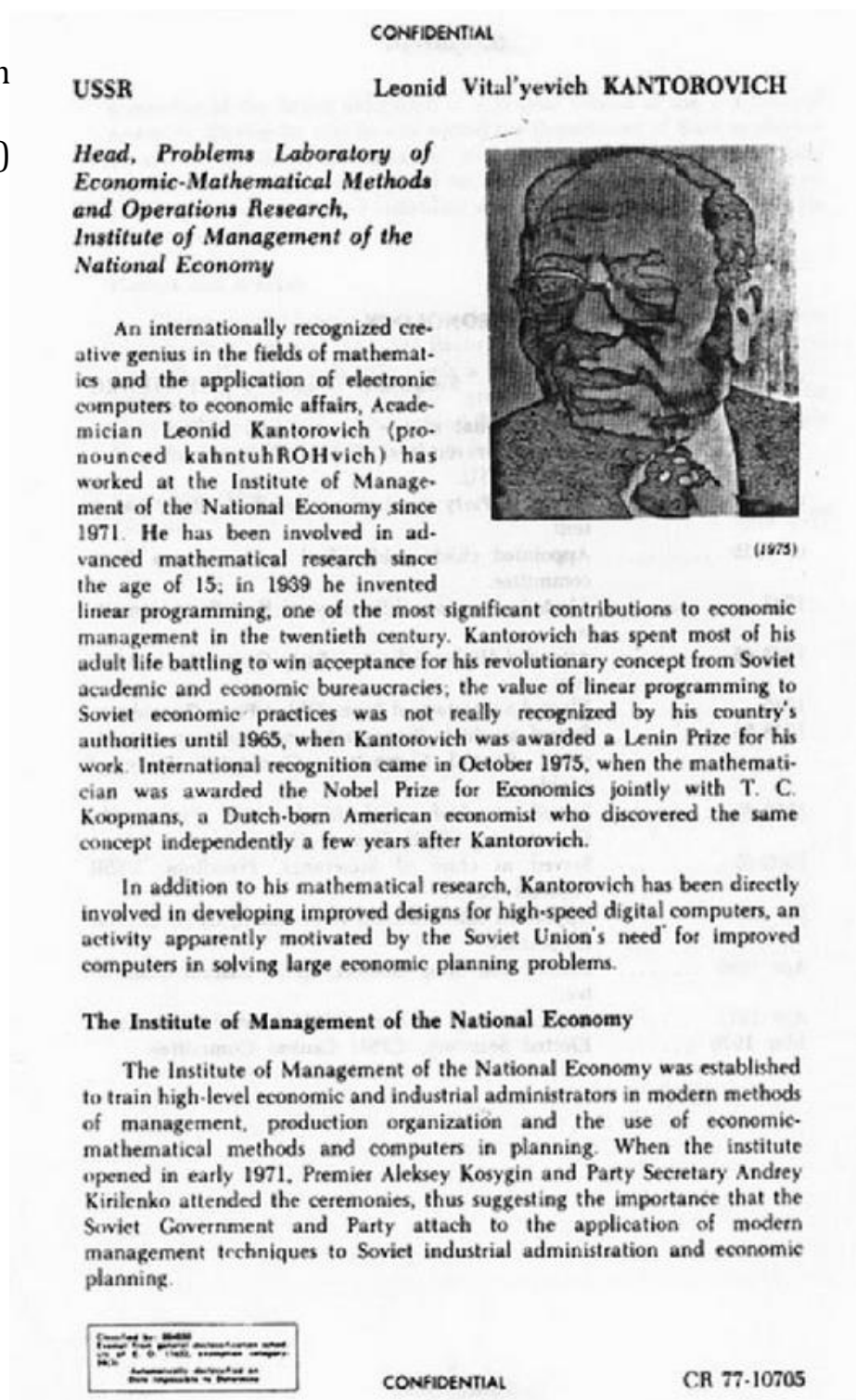
From [Bert09]:

$\kappa$	$\left(\frac{M-m}{M+m}\right)^2$	# of iterations needed to reduce optimality gap by 0.1
1.1	0.0023	1
3	0.25	2
10	0.67	6
100	0.96	58
200	0.98	116
400	0.99	231



Analysis first done by Kantorovich  
(winner of 1975 Nobel Prize in  
Economics ).

Original CIA file on  
Kantorovich, seized from  
the former US Embassy  
in Tehran. (source: Wiki)





Large  $\kappa$ :

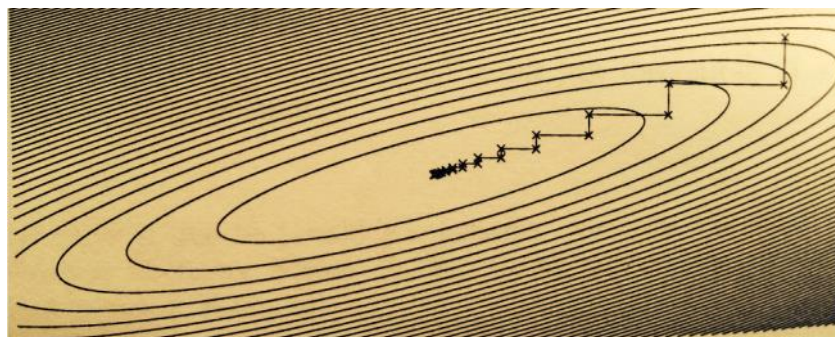


Image credit: [Bert09]

Small  $\kappa$ :



- What if the function  $f$  we are minimizing is not quadratic?
  - Denote the optimal solution by  $x_*$
  - Locally the function is well approximated by a quadratic:  
 $\frac{1}{2}x^T \nabla^2 f(x_*)x$  (plus linear and constant terms)
  - Hence  $\kappa(\nabla^2 f(x_*))$  (i.e., the condition number of the Hessian at  $x_*$ ) dictates convergence rate

We will see in future lectures how we can achieve a better than linear rate of convergence by using the Newton method.

## Lec8p18, ORF363/COS323

- The content on these last two pages has been added in response to a question raised by a student on Piazza in Fall 2015.

o How to find the order and factor of convergence?

Recall the definition:  $\{x_k\}$  converges to  $x_*$  with order  $p$  and factor  $\delta$  if

$\exists k_0$  such that  $\forall k \gg k_0$  we have

$$\|x_{k+1} - x_*\| \leq \delta \|x_k - x_*\|^p.$$

- o We want the largest  $p$  for which the above inequality holds for  $k$  large.
- o For that  $p$ , we want the smallest  $\delta$  that makes the inequality hold.
- o Here is a simple way to find  $p$  and  $\delta$ :

1) Take the following limit:  $\lim_{k \rightarrow \infty} \frac{\|x_{k+1} - x_*\|}{\|x_k - x_*\|^p}$ .

2) The order  $p$  is the largest number for which this limit is finite.

3) When you plug in that largest  $p$ , then the factor  $\delta$  is just the value of the limit.

## Lec8p19, ORF363/COS323

o Here are three examples.

$$\textcircled{1} \left\{ \frac{1}{k} \right\} \quad \left\{ 1, \frac{1}{2}, \frac{1}{3}, \frac{1}{4}, \frac{1}{5}, \dots \right\}, \quad x_* = 0$$

$$\lim_{k \rightarrow \infty} \frac{\|x_{k+1} - x_*\|}{\|x_k - x_*\|^p} = \lim_{k \rightarrow \infty} \frac{\frac{1}{k+1}}{\frac{1}{k^p}} = \lim_{k \rightarrow \infty} \frac{k^p}{k+1} = \begin{cases} \infty, & \text{if } p > 1 \\ 1, & \text{if } p = 1 \end{cases}$$

$$\Rightarrow p=1, \quad \delta=1.$$

$$\textcircled{2} \{a^k\} \text{ for } 0 < a < 1. \text{ Say } a = \frac{1}{2}: \left\{ \frac{1}{2^k} \right\} \quad \left\{ \frac{1}{2}, \frac{1}{4}, \frac{1}{8}, \frac{1}{16}, \frac{1}{32}, \dots \right\}, \quad x_* = 0.$$

$$\lim_{k \rightarrow \infty} \frac{\|x_{k+1} - x_*\|}{\|x_k - x_*\|^p} = \lim_{k \rightarrow \infty} \frac{\frac{1}{2^{k+1}}}{\left(\frac{1}{2^k}\right)^p} = \lim_{k \rightarrow \infty} \frac{2^{kp}}{2 \cdot 2^k} = \begin{cases} \infty, & \text{if } p > 1 \\ \frac{1}{2}, & \text{if } p = 1 \end{cases}$$

$$\Rightarrow p=1, \quad \delta = \frac{1}{2}.$$

$$\textcircled{3} \left\{ \frac{1}{2^{2^k}} \right\} \quad \left\{ \frac{1}{4}, \frac{1}{16}, \frac{1}{256}, \frac{1}{65536}, \dots \right\}, \quad x_* = 0.$$

$$\lim_{k \rightarrow \infty} \frac{\|x_{k+1} - x_*\|}{\|x_k - x_*\|^p} = \lim_{k \rightarrow \infty} \frac{\frac{1}{2^{2^{k+1}}}}{2^{p \cdot 2^k}} = \lim_{k \rightarrow \infty} \frac{2^{p \cdot 2^k}}{2^{2 \cdot 2^k}} = \begin{cases} \infty, & \text{if } p > 2 \\ 1, & \text{if } p = 2 \end{cases}$$

$$\Rightarrow p=2, \quad \delta=1.$$

### Notes:

- The relevant chapter of [CZ13] for this lecture is Chapter 8. You can have a look at some examples that are worked out there in detail, but if you understand the notes well, that should be enough.

### References:

- [Bert09] D. Bertsimas. Lecture notes on optimization methods (6.255). MIT OpenCourseWare, 2009.
- [Bert03] D.P. Bertsekas. Nonlinear Programming. Second edition. Athena Scientific, 2003.
- [CZ13] E.K.P. Chong and S.H. Zak. An Introduction to Optimization. Fourth edition. Wiley, 2013.
- [Tit13] A.L. Tits. Lecture notes on optimal control. University of Maryland, 2013.