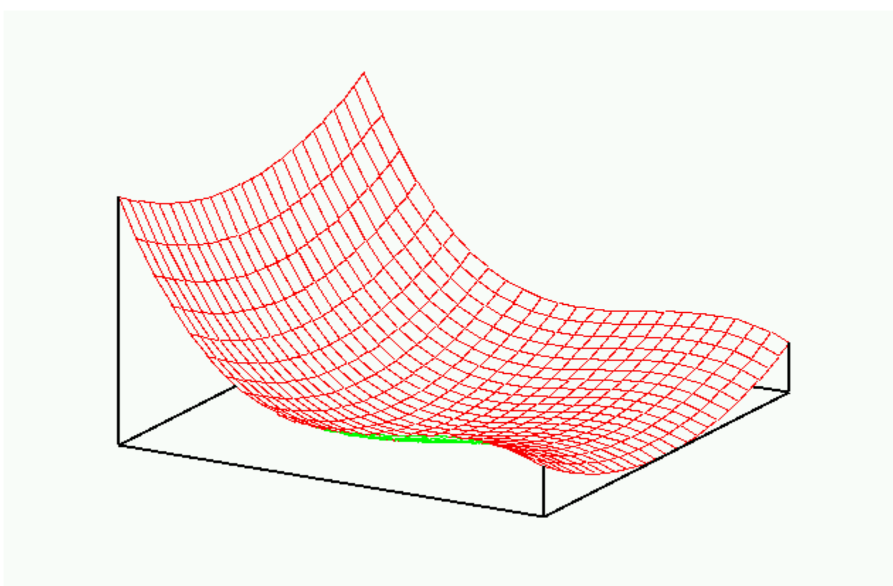
Part 3

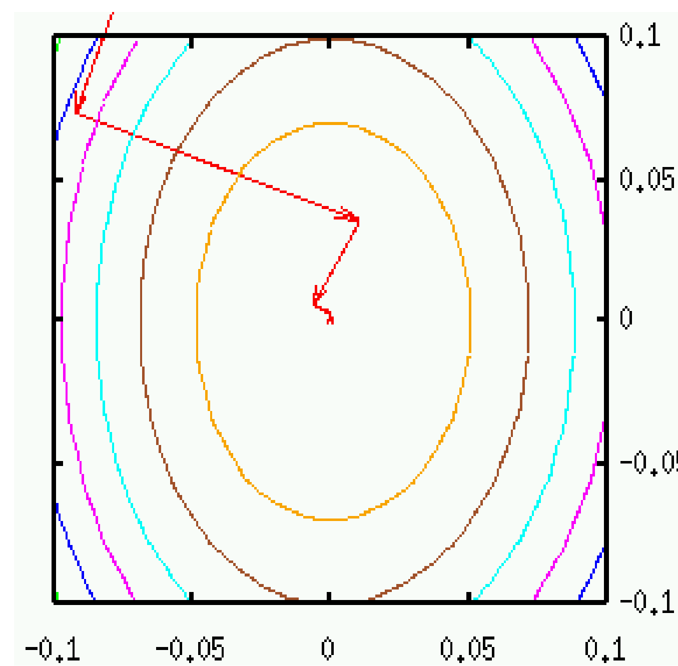
Metrics of algorithmic complexity

Outline of optimization algorithms

All algorithms to find minima of *f(x)* do so iteratively:

- start at a point x_0
- for *k=1,2,...*, :
    - . compute an update direction p_k
    - . compute a step length α_k
    - . set $x_k \leftarrow x_{k-1} + \alpha_k \, p_k$
    - . set $k \leftarrow k+1$





Outline of optimization algorithms

All algorithms to find minima of *f(x)* do so iteratively:

- start at a point x_0
- for *k=1,2,...*, :
  - . compute an update direction p_k
  - . compute a step length α_k
  - . set $x_k \leftarrow x_{k-1} + \alpha_k \, p_k$
  - . set $k \leftarrow k+1$

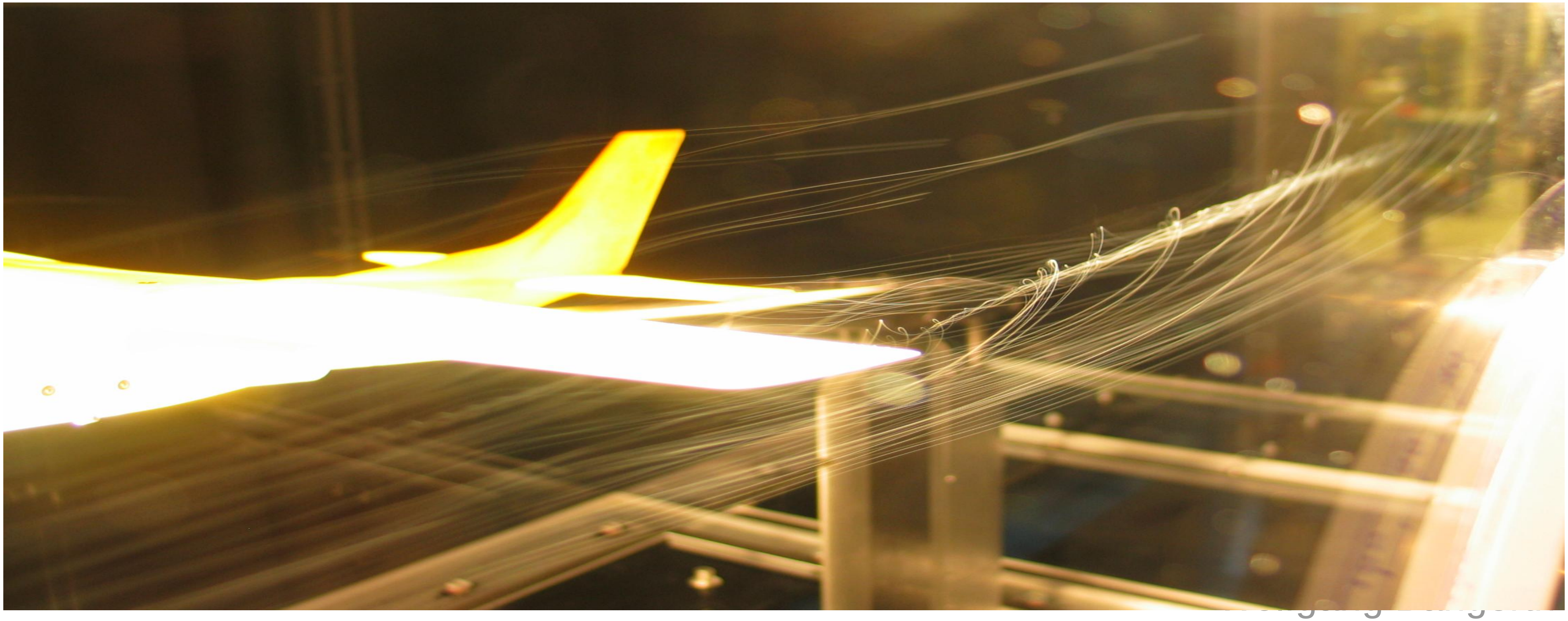
Questions:

- If x^* is the minimizer that we are seeking,
  does $x_k \rightarrow x^*$?
- How many iterations does it take for $\|x_k - x^*\| \leq \epsilon$?
- How expensive is every iteration?

How expensive is every iteration?

The cost of optimization algorithms is dominated by evaluating *f(x), g(x), h(x)* and derivatives:

- **Traffic light example:** Evaluating *f(x)* requires us to sit at an intersection for an hour, counting cars
- **Designing air foils:** Testing an improved wing design in a wind tunnel costs millions of dollars.



How expensive is every iteration?

Example: Boeing wing design

Boeing 767 (1980s)

50+ wing designs tested in wind tunnel

Boeing 777 (1990s)

18 wing designs tested in wind tunnel

Boeing 787 (2000s)

10 wing designs tested in wind tunnel

Planes today are 30% more efficient than those developed in the 1970s. Optimization in the wind tunnel and *in silico* made that happen but is *very* expensive.

How expensive is every iteration?

Practical algorithms:

To determine the search direction p_k

- Gradient (steepest descent) method requires 1 evaluation of $\nabla f(\cdot)$ per iteration
- Newton's method requires 1 evaluation of $\nabla f(\cdot)$ and 1 evaluation of $\nabla^2 f(\cdot)$ per iteration

- If derivatives can not be computed exactly, they can be approximated by several evaluations of $f(\cdot)$ and $\nabla f(\cdot)$

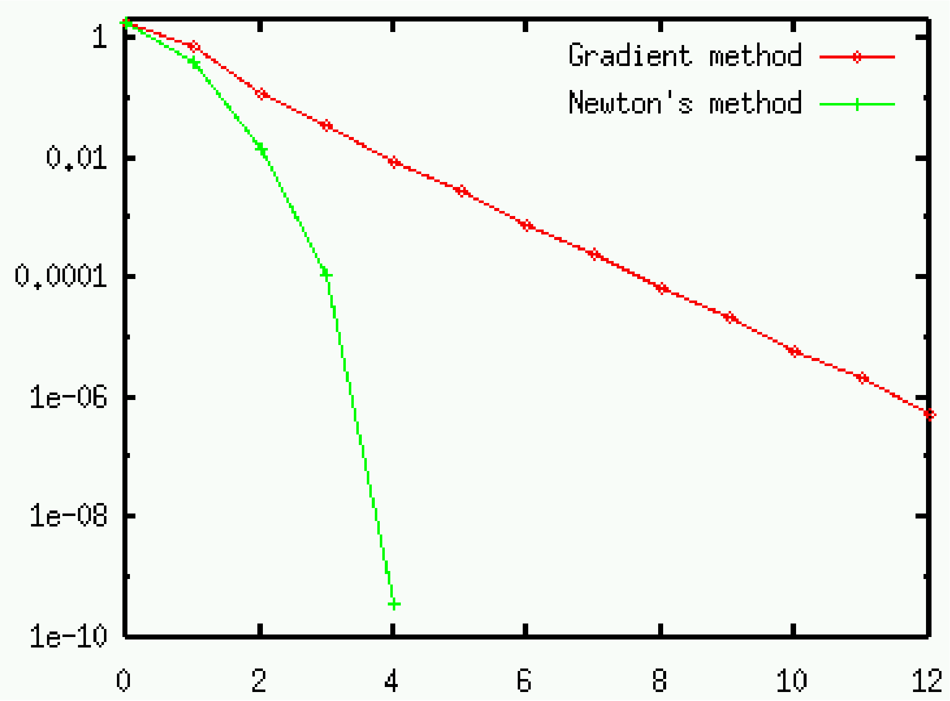
To determine the step length α_k

- Both gradient and Newton method typically require several evaluations of $f(\cdot)$ and potentially $\nabla f(\cdot)$ per iteration.

How many iterations do we need?

Question: Given a sequence $x_k \to x^*$ (for which we *know* that $\|x_k - x^*\| \to 0$), can we determine exactly *how fast the error goes to zero?*

$\|x_k - x^*\|$



k

Wolfgang Bangerth

How many iterations do we need?

Definition: We say that a sequence $x_k \to x^*$ is of order *s* if

$$\|x_k - x^*\| \leq C \|x_{k-1} - x^*\|^s$$

A sequence of numbers $a_k \to 0$ is called of order *s* if

$$|a_k| \leq C |a_{k-1}|^s$$

C is called the *asymptotic constant*. We call $C|a_{k-1}|^{s-1}$ *gain factor.*

Specifically:

If *s=1*, the sequence is called *linearly convergent*.
Note: Convergence requires *C<1*. In a singly logarithmic plot, linearly convergent sequences are straight lines.

If *s=2*, we call the sequence *quadratically convergent*.

If *1<s<2*, we call the sequence *superlinearly convergent*.

How many iterations do we need?

Example: The sequence of numbers

$$a_k = 1, 0.9, 0.81, 0.729, 0.6561, ...$$

is *linearly* convergent because

$$|a_k| \leq C |a_{k-1}|^s$$

with $s=1$, $C=0.9$.

Remark 1: Linearly convergent sequences can converge very slowly if C is close to 1.

Remark 2: Linear convergence is considered *slow.* We will want to avoid linearly convergent algorithms.

How many iterations do we need?

Example: The sequence of numbers

a_k = 0.1, 0.03, 0.0027, 0.00002187, ...

is *quadratically* convergent because

$$|a_k| \leq C|a_{k-1}|^s$$

with *s=2, C=3*.

Remark 1: Quadratically convergent sequences can converge very slowly if *C* is large. For many algorithms we can show that they converge quadratically if a_0 is small enough since then

$$|a_1| \leq C|a_0|^2 \leq |a_0|$$

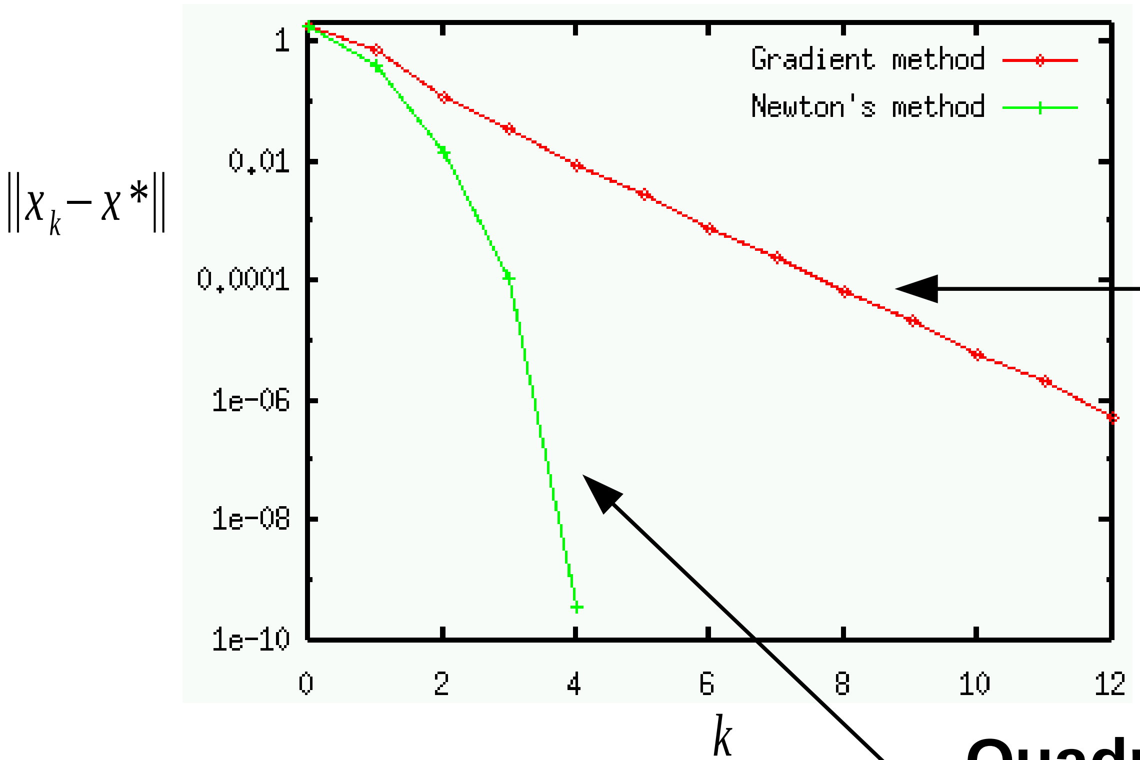
If a_0 is too large then the sequence may fail to converge since

$$|a_1| \leq C|a_0|^2 \geq |a_0|$$

Remark 2: Quadratic convergence is considered *fast.* We will want to use quadratically convergent algorithms.

How many iterations do we need?

Example: Compare linear and quadratic convergence



Linear convergence.

Gain factor C<1
is constant.

Quadratic convergence.

Gain factor $C|a_{k-1}|<1$
becomes better and better!

Metrics of algorithmic complexity

Summary:

- Quadratic algorithms converge faster *in the limit* than linear or superlinear algorithms
- Algorithms that are better than linear will need to be started *close enough* to the solution

Algorithms are best compared by counting the number of

- function,
- gradient, or
- Hessian evaluations

to achieve a certain accuracy. This is generally a good measure for the run-time of such algorithms.

Wolfgang Bangerth