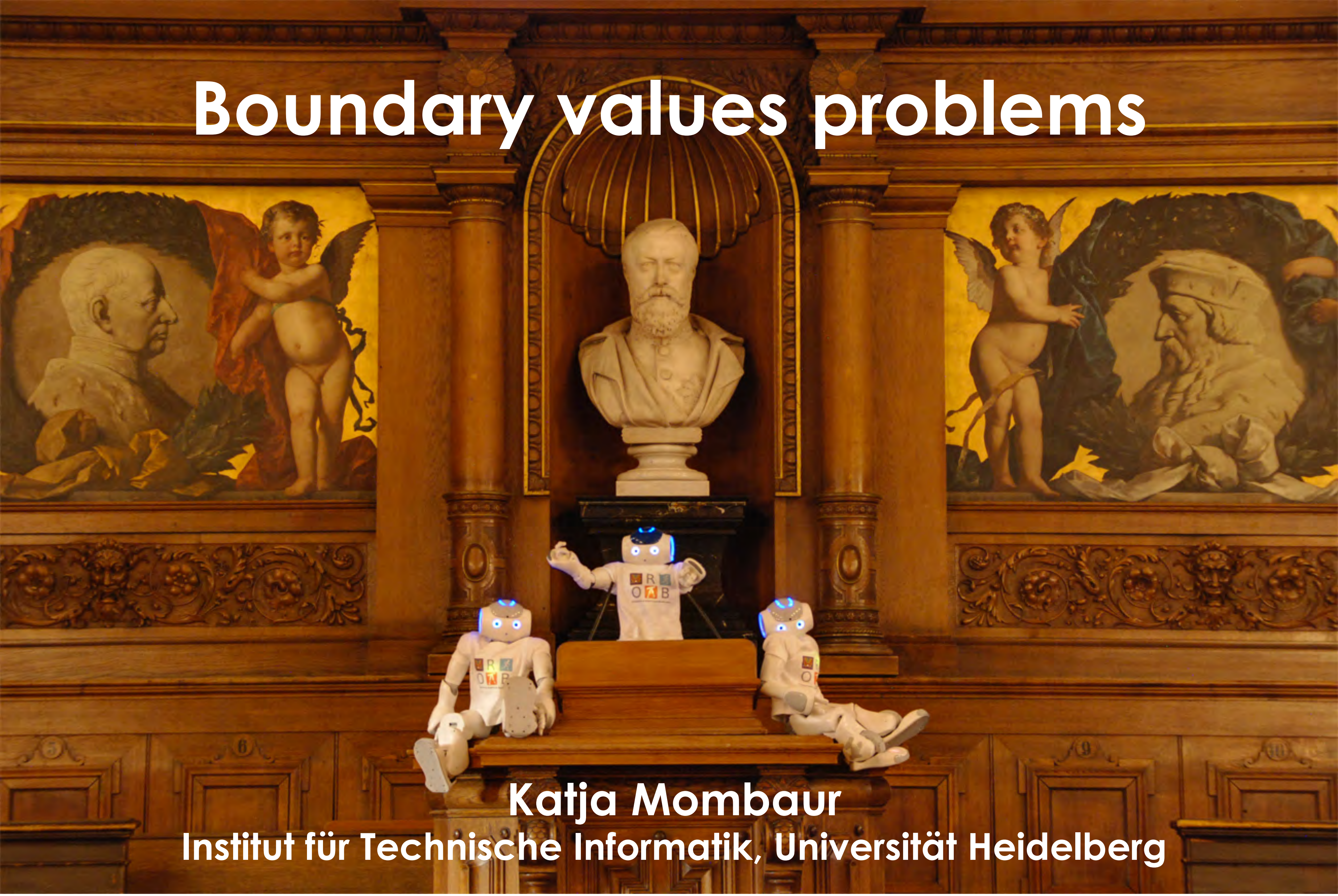
Boundary values problems



Katja Mombaur
Institut für Technische Informatik, Universität Heidelberg

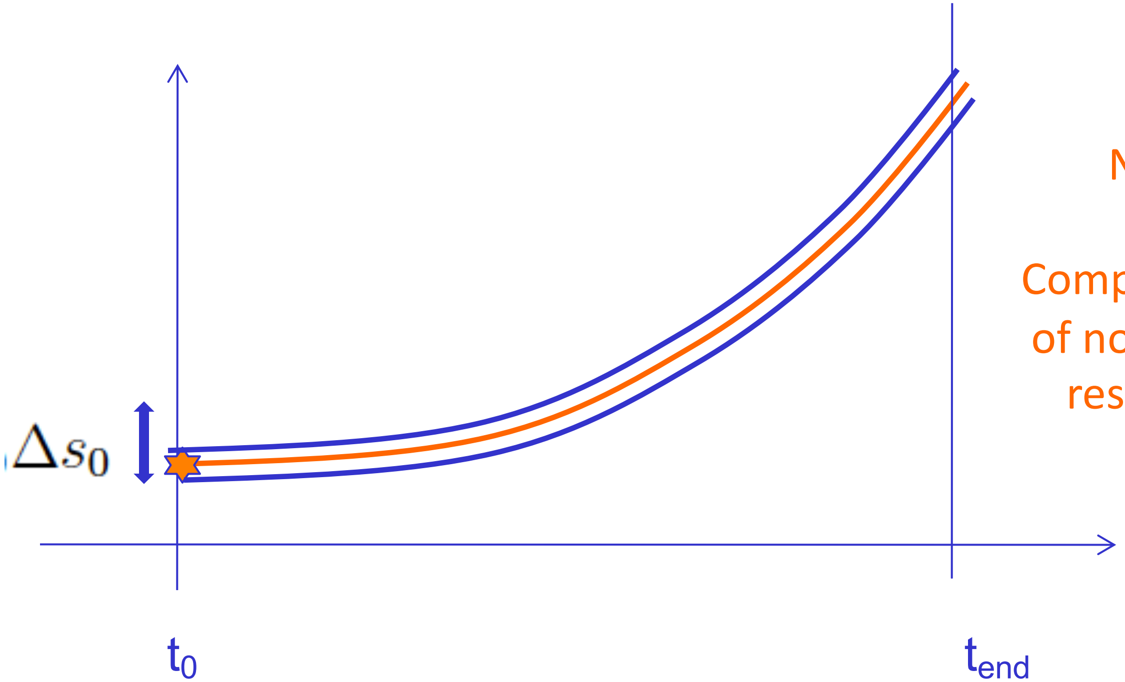
Robotics 2 - 03 June 2019

Solution methods for boundary value problems

- Last time:
  - Single Shooting
  - Multiple shooting
  - Collocation
- This time:
  - Computation of sensitivities of trajectories as required for single shooting and multiple shooting

Computation of sensitivies of trajectories

- Goal: determine sensitivity matrix that is required for single shooting and multiple shooting
- Three different approaches will be discussed here



Nominal trajectory

Computation of derivatives of nominal trajectory with respect to initial values

How would you do that?

Variant 1: Computation of sensitivies of trajectories with END

Variant 1: External numerical differentiation (END) / Variation of trajectories / Finite Differences

- Simplest, but usually also worst variant
- Apply finite difference formula for computation of derivatives

$$\frac{d}{dy} f(\bar{y}) = \frac{f(\bar{y}+h) - f(\bar{y})}{h}$$

to a variation of trajectories:

modify component i of initial value s_0 $\quad$ i = 1,....., n

$$\frac{dx_{end}}{dx_{0\,i}} = \frac{x(t_{end}; t_0, s_0 + he_i) - x(t_{end}; t_0, s_0)}{h}$$

Gives column i of matrix

Computation of sensitivies of trajectories with END

Problems:

1. Choice of perturbation of initial value h

- **Not too big**, since otherwise differences (secants) are no good approximation for derivatives (tangents)

$$\frac{dF(x)}{dx_i} = \frac{F(x+he_i) - F(x)}{h} + \mathcal{O}(h^2) \qquad (*)$$

- **Not too small**, since otherwise cancellation will occur when computing the differences
- Reminder (e.g. from optimization lecture): rule of thumb for the selection of h: h = half the digits that are used for function evaluation, i.e. for evaluation with machine epsilon: $h \sim \sqrt{\varepsilon}$
- BUT: Here the evaluation of the function F depends on the accuracy of the integration of the trajectory, i.e. for integration accuracy 10^{-TOL} the accuracy of the derivative can be in the order of $10^{-TOL/2}$ at best

Computation of sensitivies of trajectories with END

2. If this methods is applied in a stupid way, i.e. with „black box“ integrators (which in general is the case), then every integration – of every single trajectory – uses a different scheme for the integrator steps, and then additional terms add to eqn (*) describing the dependency of the step sizes on the initial values.

Remark: small improvements can of course be achieved by by using central finite differences, but the basic problems remain

Conclusion:

- Method leads to acceptable derivative information only for very high integration tolerance
- But in this case it is very expensive! Solution: next method!

Variant 2: Computation of derivaties with Internal Numerical Differentiation (IND)

H. G. Bock (Heidelberg)

Principle:

Compute the derivative of the discretization scheme that generates the solution of the base trajectory

Applied to a variation of trajectories:

- Use same step sizes scheme for integration of base trajectory and varied trajectories
- In general, not the full varied trajectories are computed (which would cause problems for instable solutions), but instead we
  - Apply a perturbation and integrate over a short interval in the extreme case: perform one single integration step
  - Compute overall sensitivity matrix via multiplication of the individual step sensitivity matrices (chain rule)

$$G_q \cdot G_{q-1} \cdots G_1$$

Cmputation of sensitivities with IND

- **Accuracy of IND:**

  – Here we compute the derivative of an analytic function – the discretization scheme – i.e. the applied perturbation is independent of the integration accuracy, and we can apply the rule of thumb ε: $h \sim \sqrt{\varepsilon}$

  – This makes it possible to compute derivatives of the trajectory at the same order of accuracy as the trajectory itself (up to the limit) $\sqrt{\varepsilon}$

Variant 3: Computation of derivatives via the variational differential equation

- The solution $G(t_{end}; t_0)$ of the variational differential equation

$$\begin{aligned} \dot{G}(t; t_0) &= f_x(t, \bar{x}(t)) \cdot G(t; t_0) \\ G(t_0; t_0) &= I \end{aligned}$$

(we assume that f_x can be computed analytically)

  theoretically gives the exact derivative matrix of the end values of the trajectory with respect to the initial values

- If however, another step size scheme is used for the integration of the variational differential equation than for the base trajectory, then error terms occur also here

- **The principle of IND applied to the variational differential equation** means : same step size control for x(t) und G(T) – this is generally the case if they are integrated together

- In this case, **the numerical solution of the variational differential equation is the exact derivative of the numerical solution of the differential equation**

Computation of derivatives via the variational differential equation

- With the principle of IND applied to the variational differential equation, derivatives can be computed up to the accuracy of the trajectory computation – up to very high integration accuracies

Optimization

Katja Mombaur
Institut für Technische Informatik, Universität Heidelberg

Robotics 2 - 03 June 2019

Contents of lecture

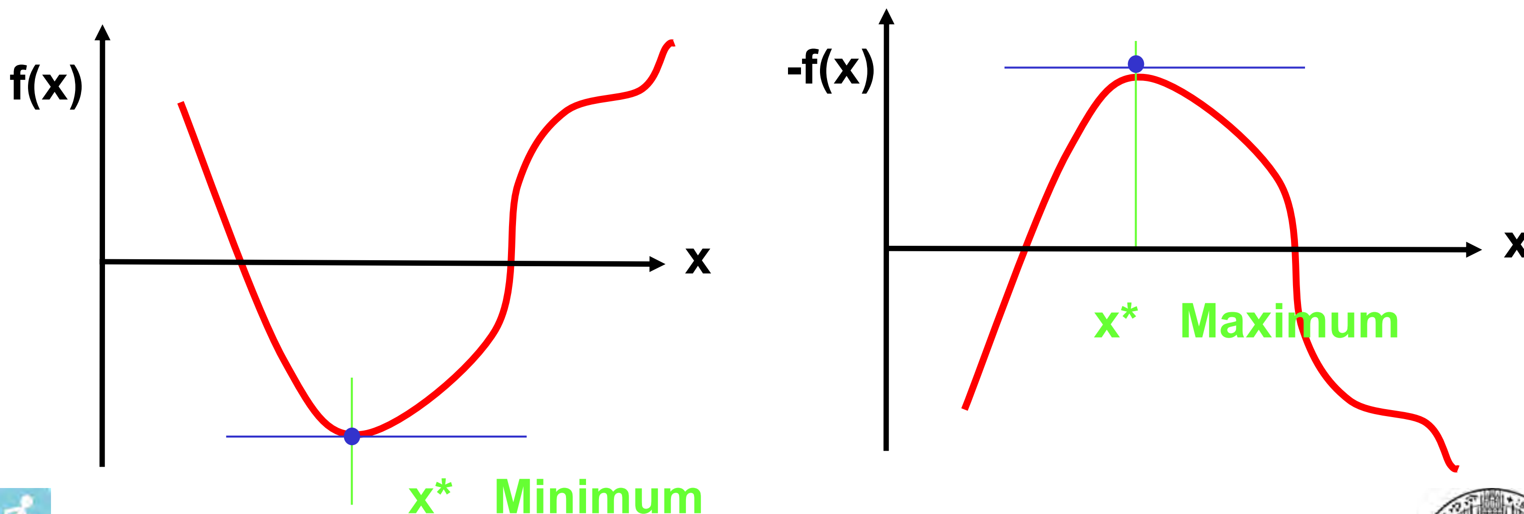
- Introduction of basic terms in optimization
- Introduction of different problem classes in optimization
- Nonlinear optimization (NLP)
- Optimality conditions
- Newton's methods / SQP methods

Optimization basics

What is optimization?

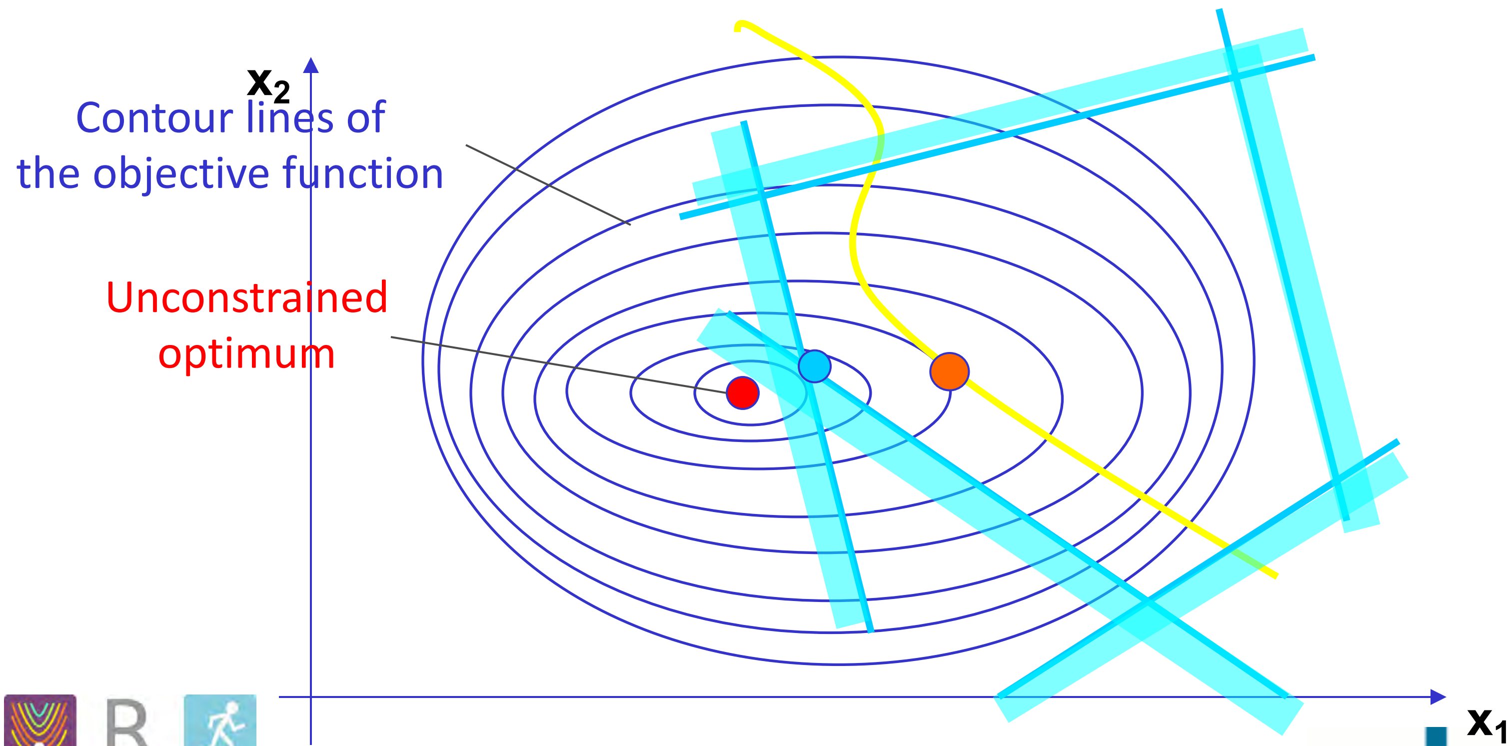
- Optimization = search for the best solution
- in mathematical terms:
  minimization or maximization of an objective function f (x) depending on variables x subject to constraints

Equivalence of minimization and maximization problems

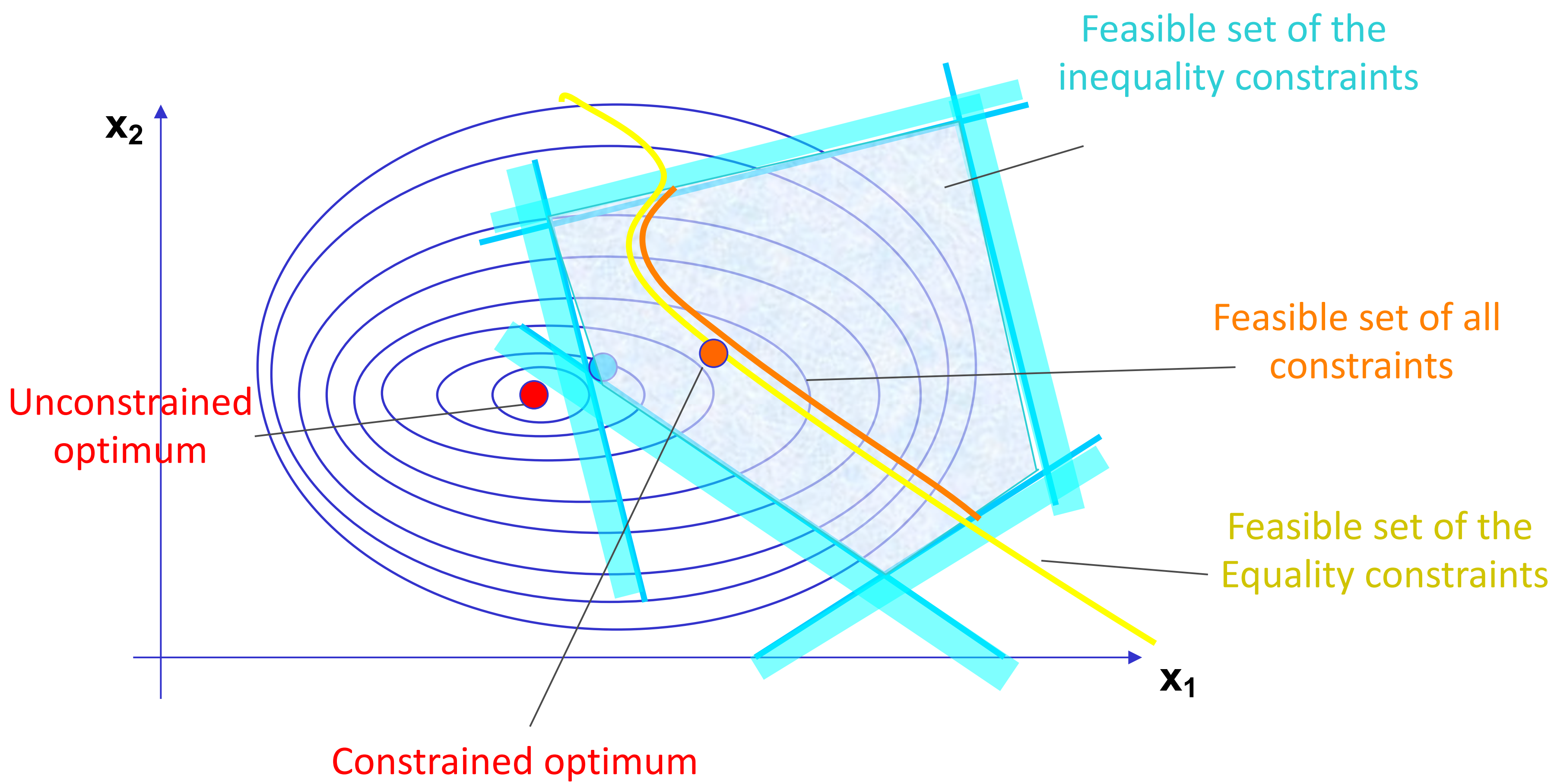


Variables & constraints

- problem depends on n independent variables $x = (x_1, x_2, \ldots, x_n)$
- in many cases, there are additional constraints
  (equality constraints g(x) and inequality constraints h(x=)

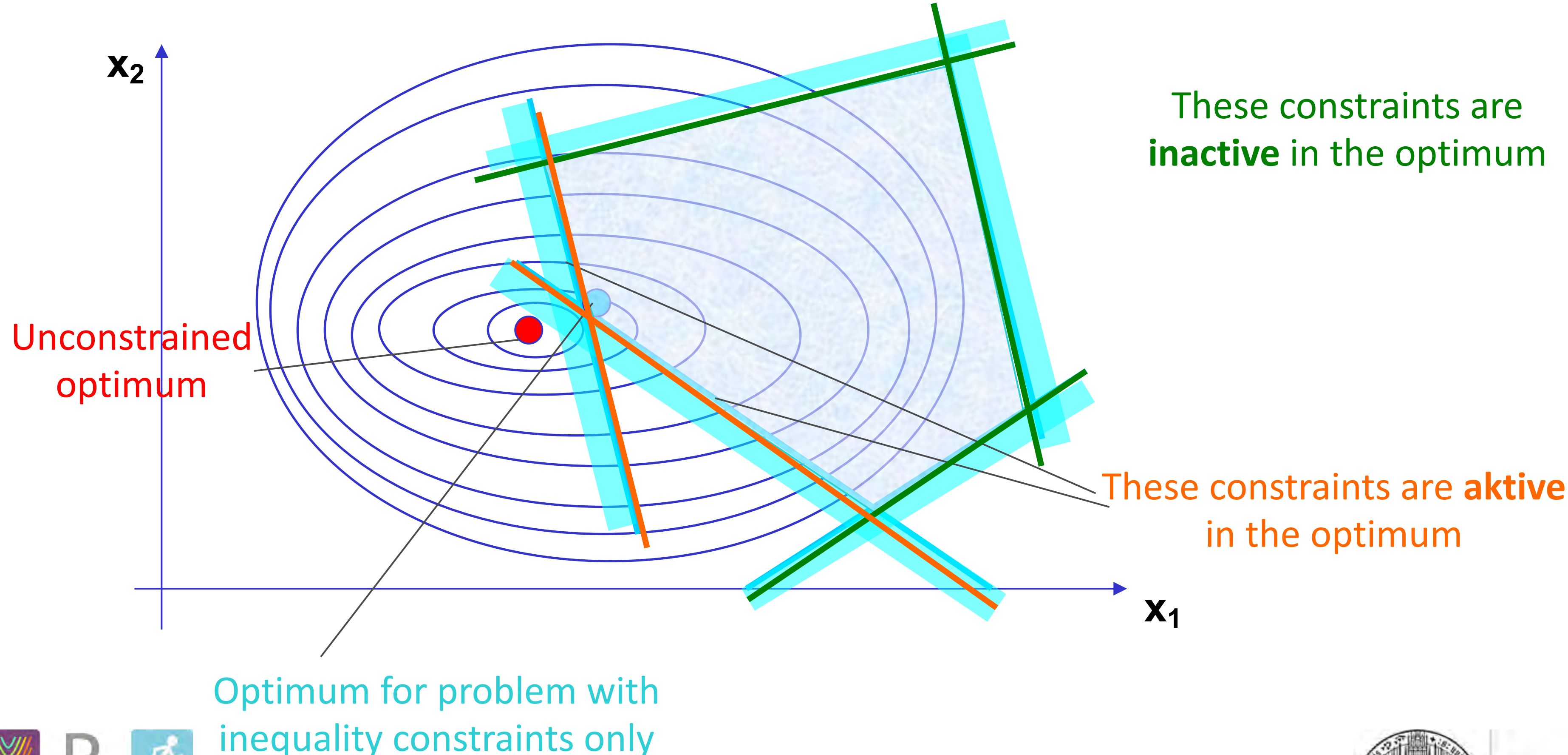


Feasible set / points

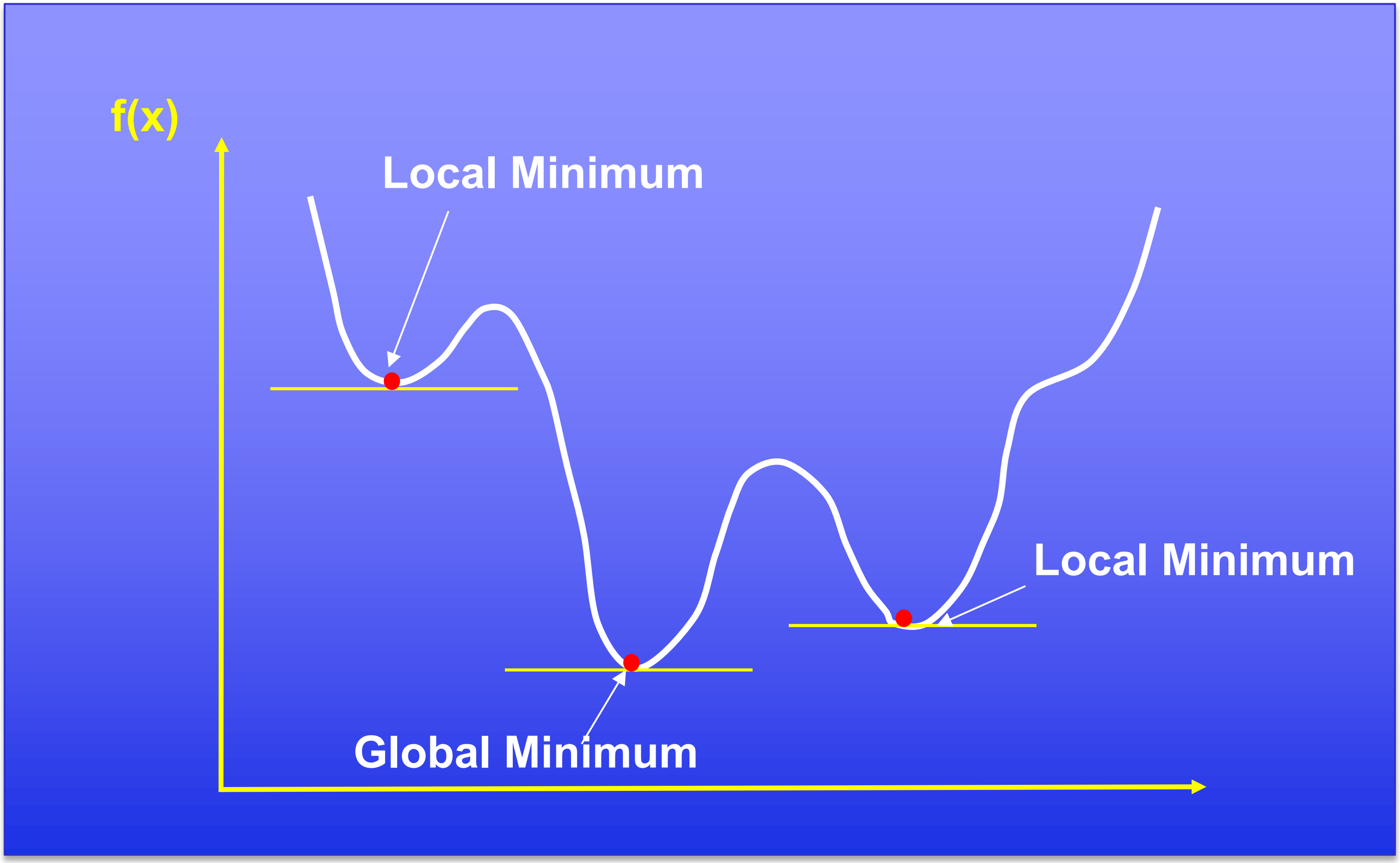


Inequalities can be active or inactive

- Here equality constraints have been omitted for clarity



Local and global optima



Derivatives

- First and second derivatives of the objective function or the constraints play an important role in optimization

- The first order derivatives are called the gradient (of the resp. fct)

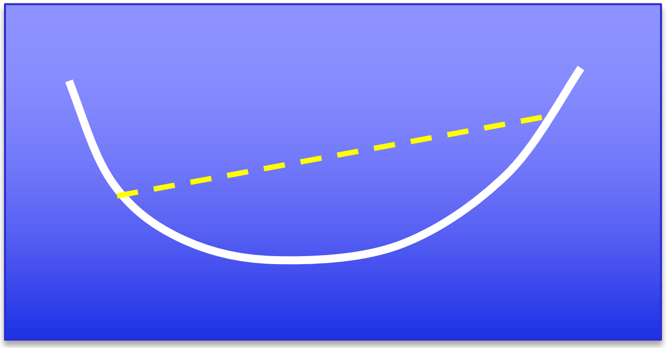
$$\nabla f(x) = \left(\frac{\partial f}{\partial x_1}, \frac{\partial f}{\partial x_2}, \ldots, \frac{\partial f}{\partial x_n}\right)^T$$

- and the second order derivatives are called the Hessian matrix

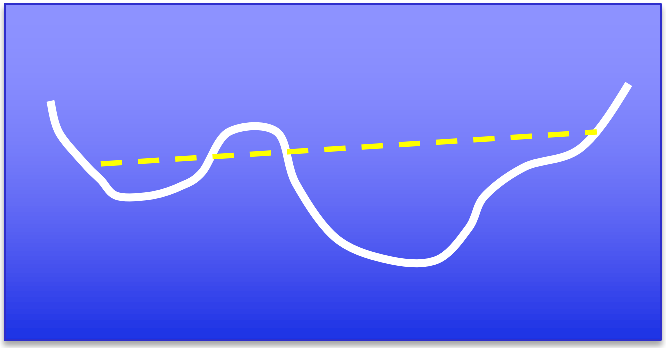
$$\nabla^2 f(x) = \begin{pmatrix} \frac{\partial^2 f}{\partial {x_1}^2} & \frac{\partial^2 f}{\partial x_1 \partial x_2} & \cdots & \frac{\partial^2 f}{\partial x_1 \partial x_n} \\ \frac{\partial^2 f}{\partial x_2 \partial x_1} & \frac{\partial^2 f}{\partial {x_2}^2} & \cdots & \frac{\partial^2 f}{\partial x_2 \partial x_n} \\ \cdots & \cdots & \cdots & \cdots \\ \frac{\partial^2 f}{\partial x_n \partial x_1} & \frac{\partial^2 f}{\partial x_n \partial x_2} & \cdots & \frac{\partial^2 f}{\partial {x_n}^2} \end{pmatrix}$$

Convex functions

- **Convex Function**
  - any line joining two points on its graph lies nowhere below graph



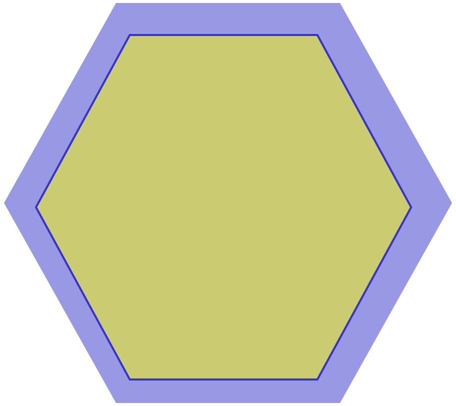
- **Nonconvex Function**
  - there exist connecting lines lying below graph



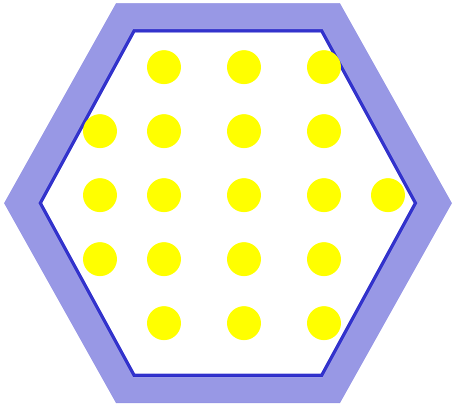
Convex functions have a single minimum!

Characteristics of optimization problems 1

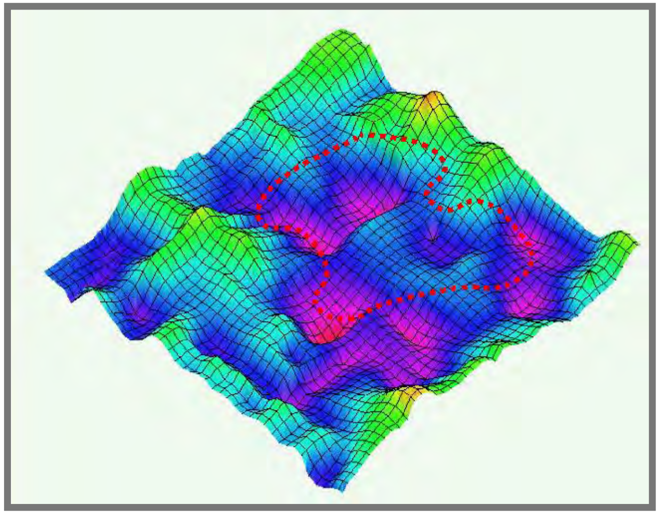
- size / dimension of problem n,
  i.e. number of free variables



- continuous or discrete search space

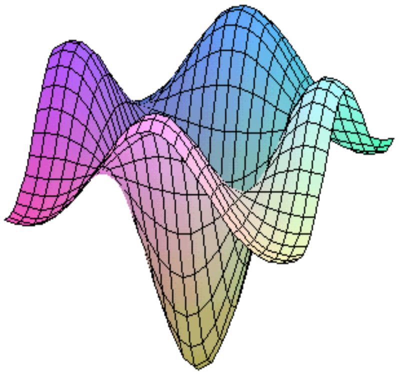


- number of minima

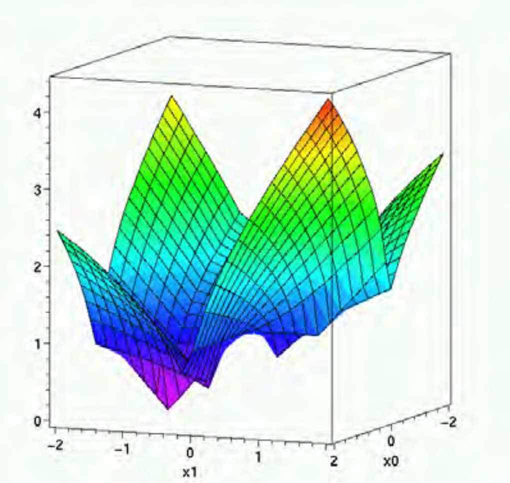


Characteristics of optimization problems 2

- Properties of the objective function:
  - type: linear, nonlinear, quadratic ...
  - smoothness: continuity, differentiability



- Existence of constraints



- Properties of constraints:
  - equalities / inequalities
  - type:
    - „simple bounds“, linear, nonlinear
    - Dynamics (ODE, DAE, PDE)
  - smoothness

Classes of Optimization problems

Classes of Optimization Problems: LP

- Linear Optimization Problem (LP)

$$\begin{array}{ll} \min\limits_{x} & c^T x \\ \text{s. t.} & Ax = b \\ & x \geq 0 \end{array}$$

- Example: Logistics Problem
  - shipment of quantities $a_1, a_2, \dots a_m$ of a product from m locations
  - to be received at n destinations in quantities $b_1, b_2, \dots b_n$
  - shipping costs c_{ij}
  - determine amounts x_{ij}

Origin of linear programming in 2nd world war

Classes of Optimization Problems: QP

- Quadratic Optimization Problem (QP)

$$
\begin{aligned}
\min_{x} \quad & c^T x + \frac{1}{2} x^T Q x \\
\text{s. t.} \quad & Ax = b \\
& Cx \geq d
\end{aligned}
$$

- Example: Markovitz mean variance portfolio optimization
  - quadratic objective: portfolio variance (sum of the variances and covariances of individual securities)
  - linear constraints specify a lower bound for portfolio return
- QPs play an important role as **subproblems in nonlinear optimization**

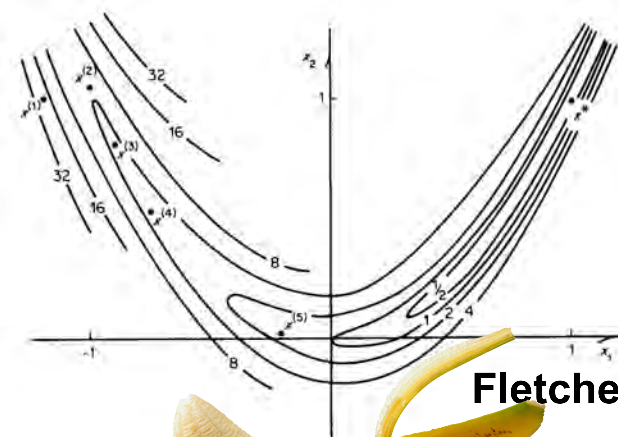
Classes of Optimization Problems NLP

- Nonlinear Optimization Problem (NLP)

$$\begin{aligned} \min_{x} \quad & f(x) \\ \text{s. t.} \quad & h(x) = 0 \\ & g(x) \geq 0 \end{aligned}$$

- Famous nonlinear example function
  - Rosenbrock function

$$f(x) = 100(x_2 - x_1^2)^2 + (1 - x_1)^2$$



Fletcher

„Banana valley Function“

Classes of Optimization Pbs. – Non-smooth optimization

- Non-Smooth Optimization Problem
  - objective function or constraints are non-differentiable or not continuous

$$f(x) = |x|$$

$$f(x) = \max_i f_i(x), \quad i = 1, ..n$$

$$f(x) = \begin{cases} \cos x & \text{für } x \leq \frac{\pi}{2} \\ 0 & \text{für } x > \frac{\pi}{2} \end{cases}$$

$$f(x) = i \quad \text{for} \quad i \leq x < i+1, \; i = 0, 1, 2, \ldots$$

Classes of Optimization Problems – (mixed) integer

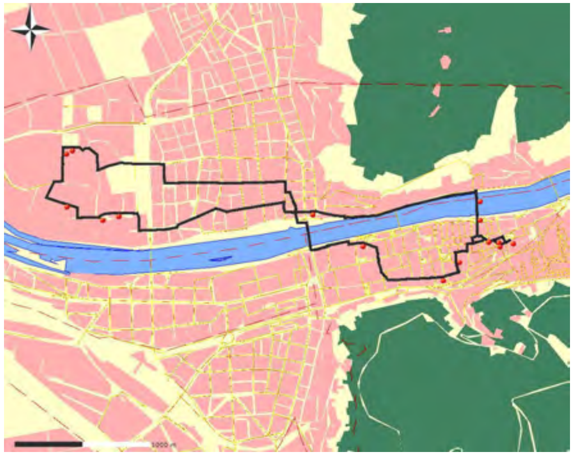
- Integer optimization problems
(e.g. linear integer problems) or

$$\begin{array}{ll} \min\limits_{x} & c^T x \\ \text{s. t.} & Ax = b \\ & x \in Z_+^n \end{array}$$

- Mixed integer problems

- Special case: combinatorial optimization problems -- feasible set is finite

- Example: traveling salesman problem
    - determine fastest/shortest round trip through n locations



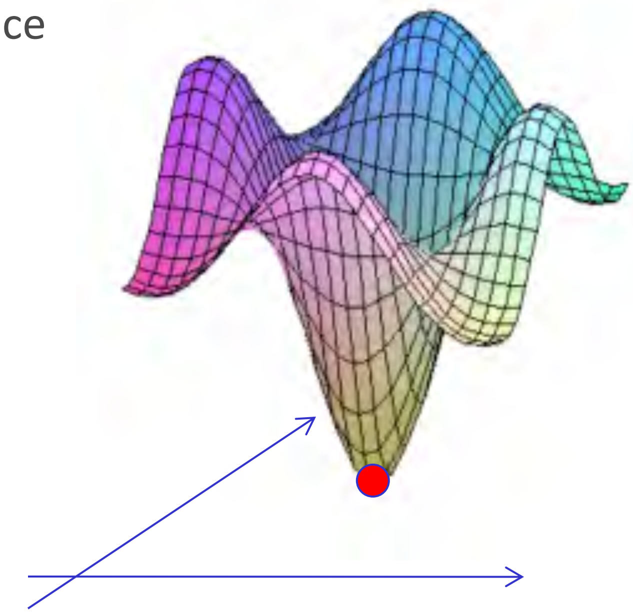
Common property of all optimization problems listed so far...

- Optimization variables are from some finite dimensional space

$$\text{e.g.} \quad x \in \mathbb{R}^n$$

- The result is a point in finite dimensional space

$$x^* = \begin{pmatrix} x_1^* \\ x_2^* \\ x_3^* \\ .. \\ x_n^* \end{pmatrix}$$



- This will change for the next classes of problems!

Optimal control problems

Optimal control = Optimal choice of inputs for a dynamic system

System dynamics can be manipulated by controls and parameters:

$$\boxed{\dot{x}(t) = f(t, x(t), u(t), p)}$$

- simulation interval: $[t_0, t_{\mathrm{end}}]$
- time $t \in [t_0, t_{\mathrm{end}}]$
- state $x(t) \in \mathbb{R}^{n_x}$
- controls $u(t) \in \mathbb{R}^{n_u}$ $\longleftarrow$ manipulated
- design parameters $p \in \mathbb{R}^{n_p}$ $\longleftarrow$ manipulated

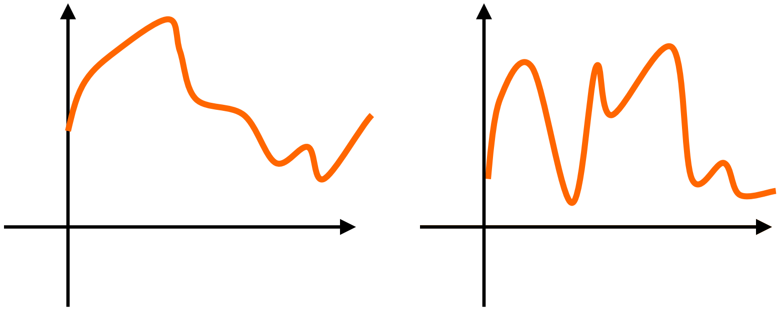
Optimal control problems

- Optimization problems **including dynamics** e.g. optimal control problems

$$
\begin{aligned}
\min \quad & \int_{t_0}^{t_{end}} \Phi(t, x(t), u(t), p)dt \\
s.t. \quad & \dot{x} = f(t, x(t), u(t), p) \\
& x(t_0) = x_0, \; x(t_{end}) = x_{end} \\
& \dots
\end{aligned}
$$

- State and control variables are functions in time

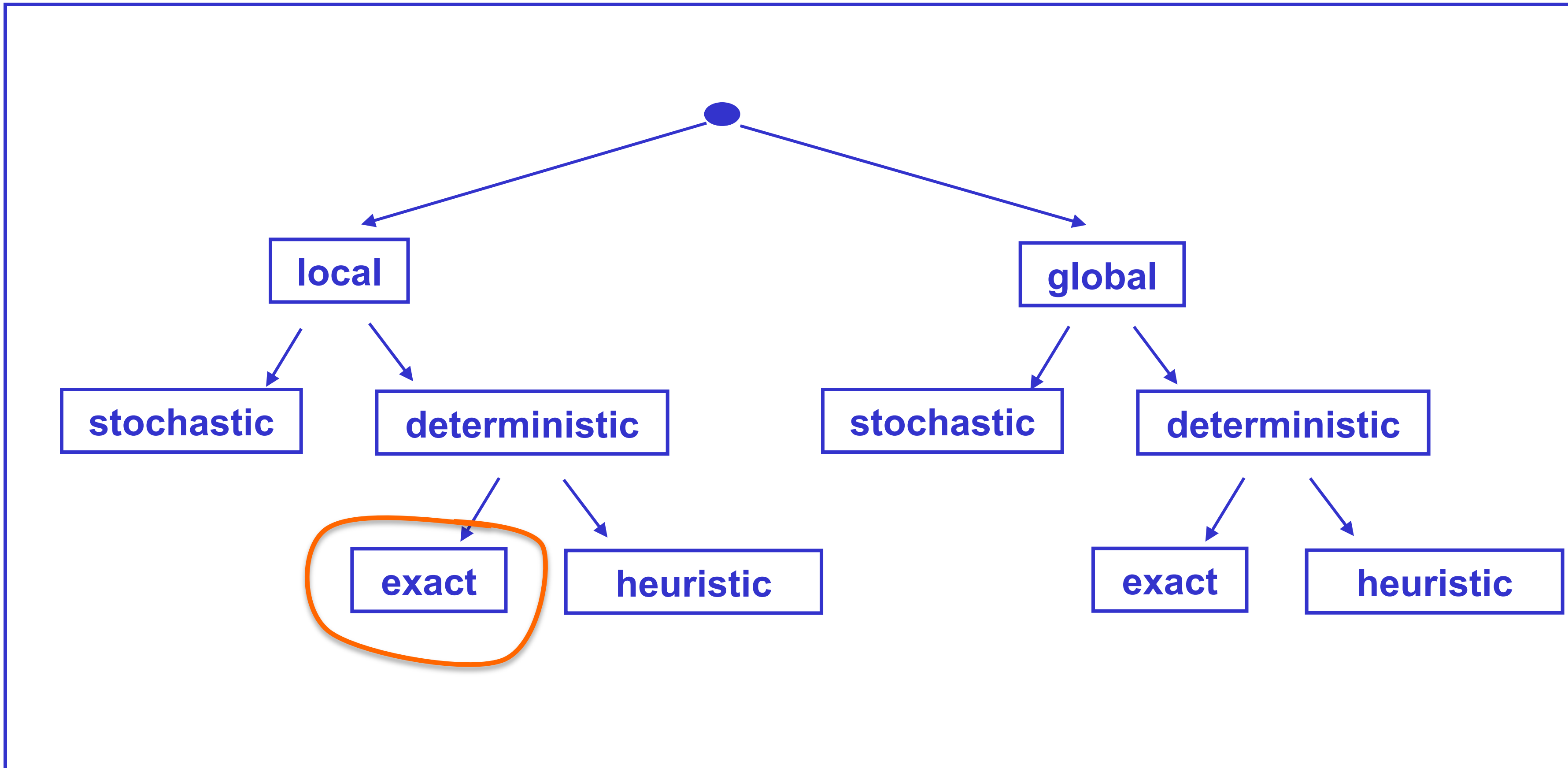
  (**infinite-dimensional** variables)



Will be discussed next week!
Now we go back to finite-dimensional NLPs

Optimization algorithms

Classification of Optimization Algorithms



Nonlinear optimization problems / Nonlinear Programming Problems (NLP)

$$\boxed{\begin{array}{rcl@{\quad}l} \min f(x) & & & f: \; D \subset R^n \to R \\ g(x) & = & 0 & g: \; D \subset R^n \to R^l \\ h(x) & \geq & 0 & h: \; D \subset R^n \to R^k \end{array}}$$

f(x) objective function / cost function

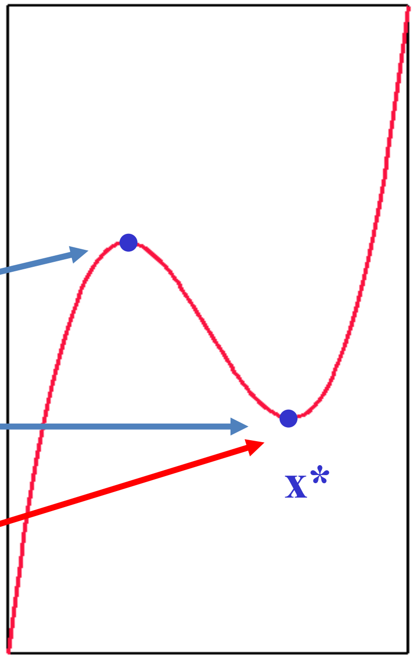
g(x) - equality constraints

h(x) - inequality constraints

Optimality conditions (unconstrained case)

Assumption: f is twice continuously differentiable

We want to test if a given point is a minimum



- Necessary condition:
  $\nabla f(x^*)=0$ (stationary Punkt)

- Sufficient condition:
  x^* is a statioary point and $\nabla^2 f(x^*)$ ist positive definite

Different types of stationary points

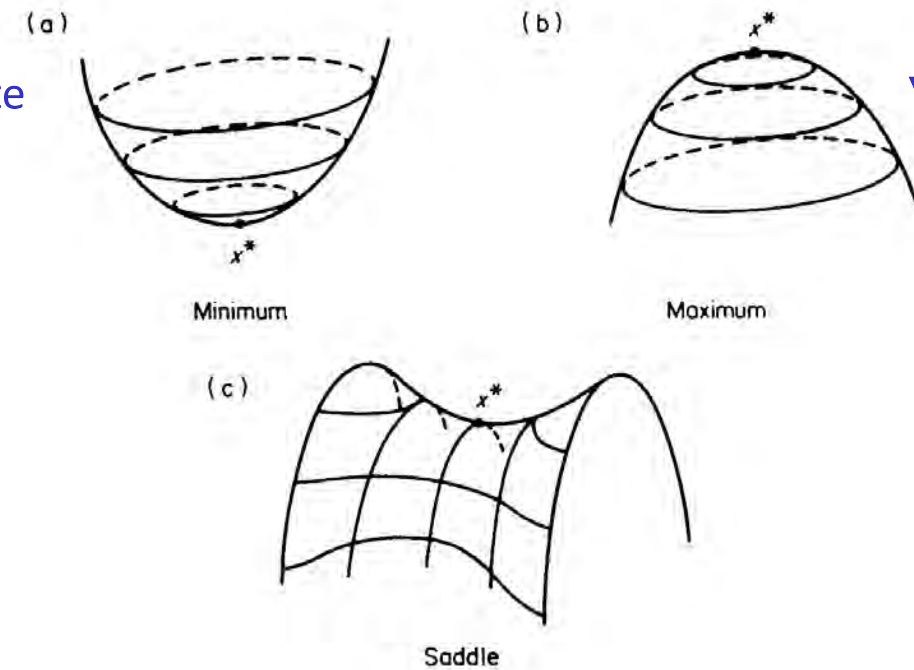
(a)-(c) x^* is stationary point $\nabla f(x^*)=0$

$\nabla^2 f(x^*)$ positive definite

Minimum

$\nabla^2 f(x^*)$ negative definite

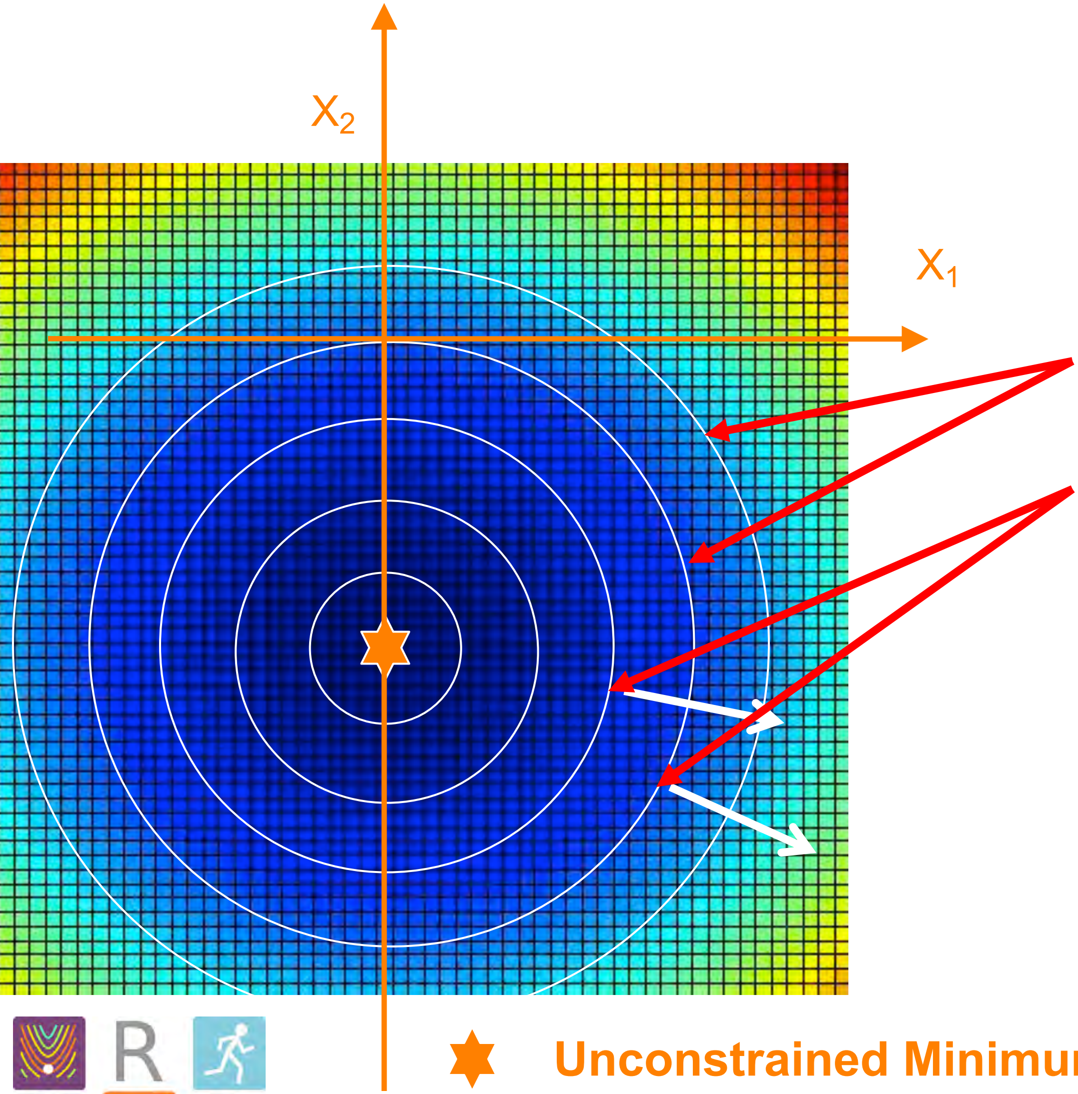
Maximum



$\nabla^2 f(x^*)$ indefinite

Saddle point

Example problem - unconstrained



$$\boxed{\min_{x \in R^2} x_1^2 + x_2^2 + m x_2}$$

Contour lines of f(x)

Gradient vector

$$\nabla f(x) = (2x_1, 2x_2 + m)$$

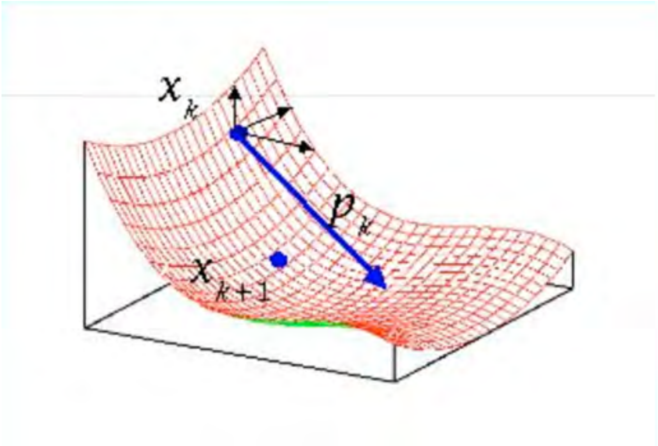
Unconstrained minimum

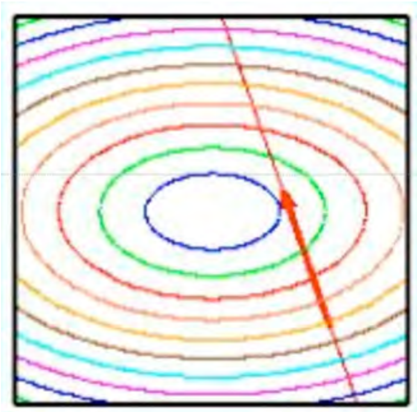
$$0 = \nabla f(x^*) \Leftrightarrow (x_1^*, x_2^*) = (0, -\frac{m}{2})$$

Unconstrained Minimum

Basic idea of iterative optimization algorithms

- Start in a point x_0
- Perform the following iteration:
  - Determine a **direction of descent p_k**
  - Determine the **step length α_k**
  - Go to the next iterate $x_{k+1} = x_k + \alpha_k p_k$





Essential difference between the different optimization algorithms:

Computation of p_k and α_k

Computation of the direction of search

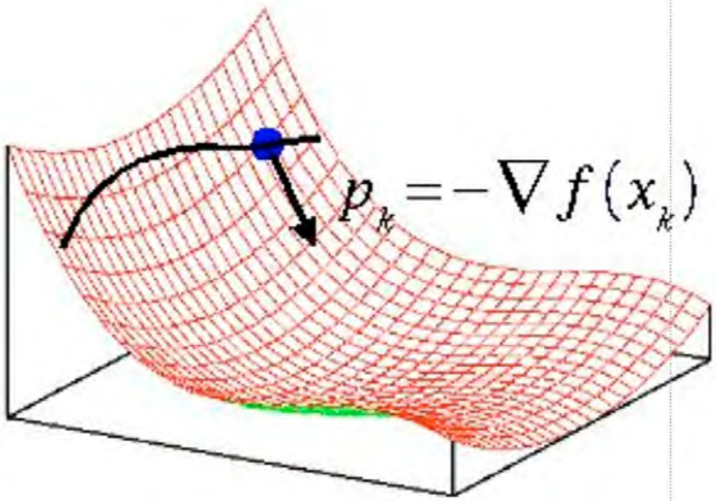
- For the determination of p frequently first and second order derivatives of f are used

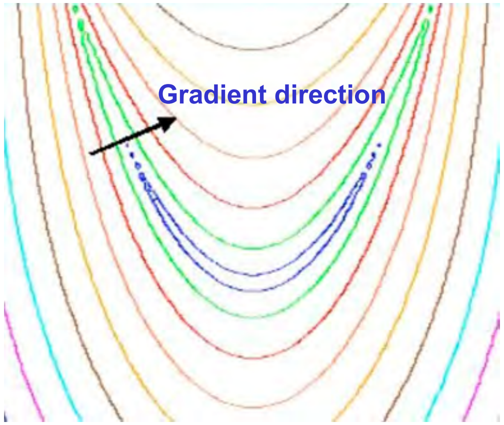
- Examples:
  - Steepest descent method
  - Newton method
  - Quasi-Newton
  - ...

Steepest descent method

Always chose the direction that provides the steepest descent

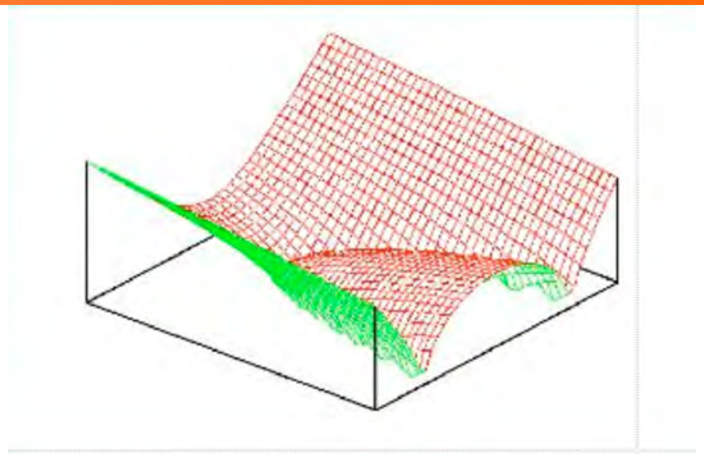
$$\boxed{p^k = -\nabla f(x^k)} \qquad \boxed{x^{k+1} = x^k - \alpha^k \nabla f(x^k)}$$

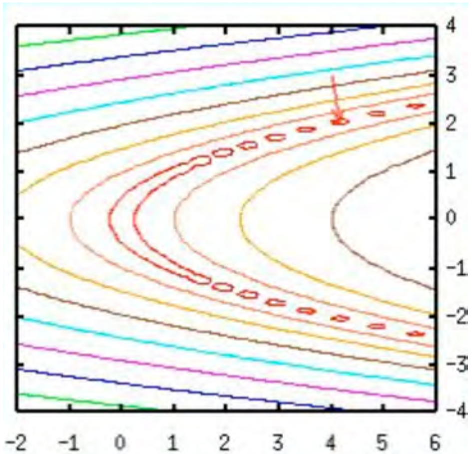


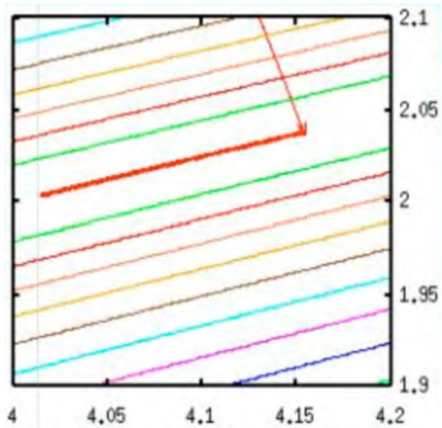


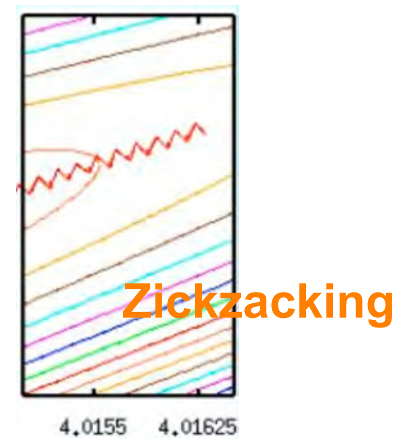
Example steepest descent method

- Behavior for banana valley function









Convergence rate of steepest descent method is very slow (linear)

Reminder (Numerik 0) – Convergence rates

- A sequence (an iterative algorithm) **converges linearly**, if:

$$|x_t - z| \leq c|x_{t-1} - z| \qquad c < 1$$

where z is the solution, x_t and x_{t-1} are the iterates, and c is a constant

- An iterative algorithm is **converges superlinearly**, if

$$|x_t - z| \leq c_t|x_{t-1} - z| \quad \text{with} \quad c_t \rightarrow 0 \ (t \rightarrow \infty)$$

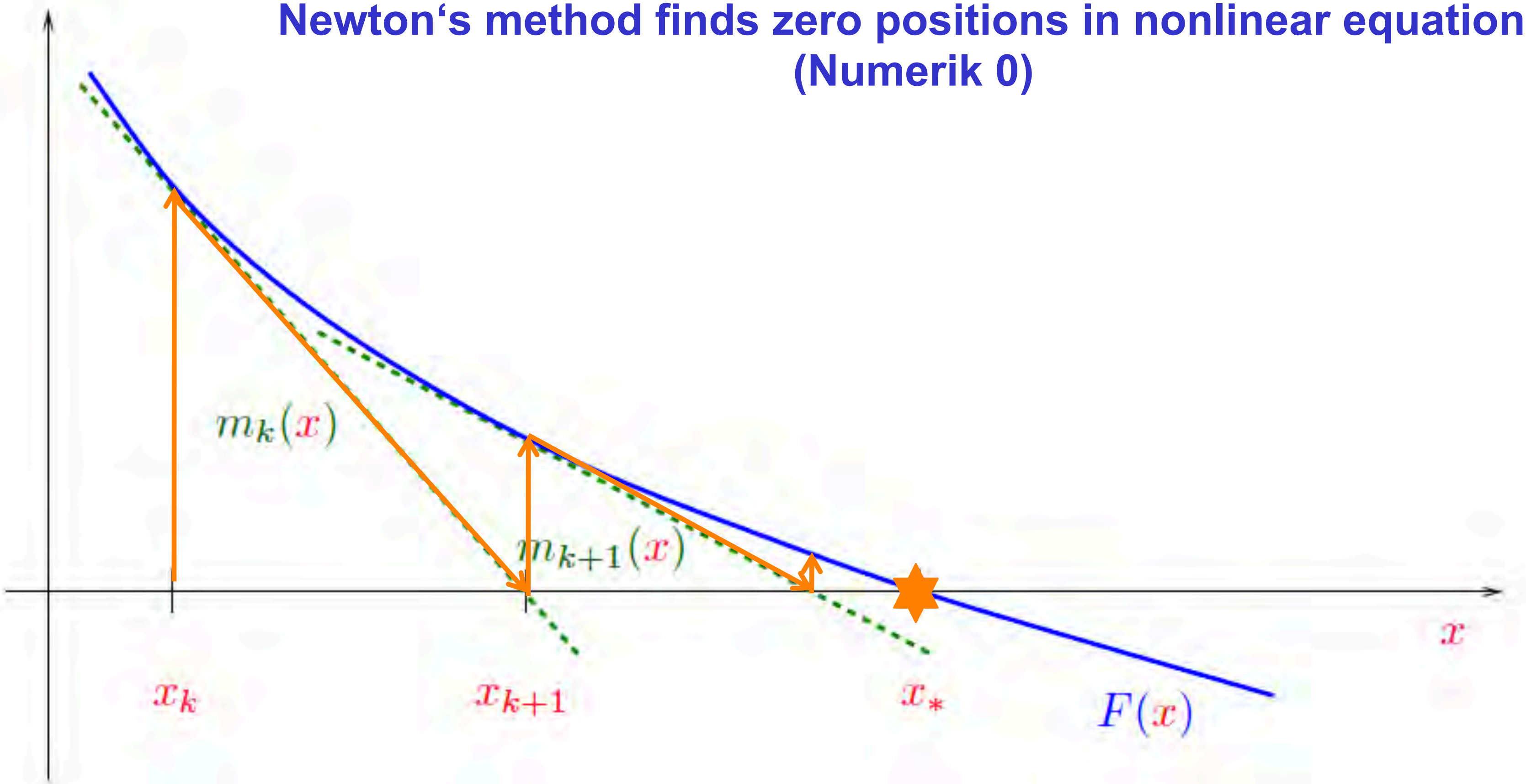
i.e. c_t is a null sequence

- An iterative algorithm is **converges quadraticly**, if

$$|x_t - z| \leq c|x_{t-1} - z|^2$$

A better idea (than steepest descent): Newton‘s method

Newton‘s method finds zero positions in nonlinear equations (Numerik 0)



How can we use that in optimization?

Newton's method for optimization

Idea: Compute the zero of the equation $\boxed{F(x) = \nabla f(x) = 0}$

in order to satisfy the first order optimaliy conditions

Taylor series

$$F(x^{k+1}) = F(x^k) + \frac{d}{dx}F(x^k)(x^{k+1} - x^k) + \ldots = 0$$

$$\Rightarrow x^{k+1} = x^k - \underbrace{\left(\frac{d}{dx}F(x^k)\right)^{-1} F(x^k)}_{p^k}$$

Newton iteration:

$$\boxed{x^{k+1} = x^k - \left(\nabla^2 f(x^k)\right)^{-1} \nabla f(x^k)}$$

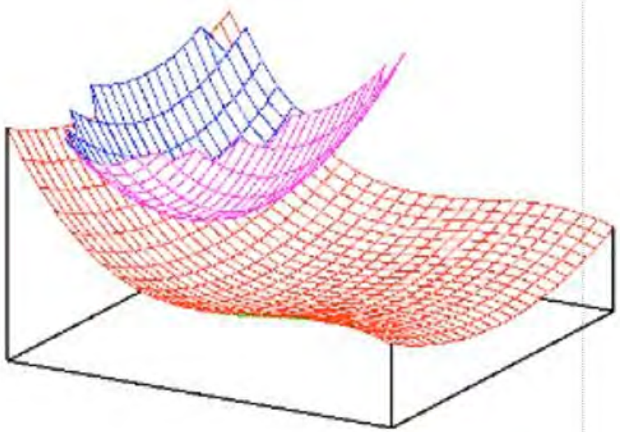
inverse of the Hessian matrix gradient
of the objective function

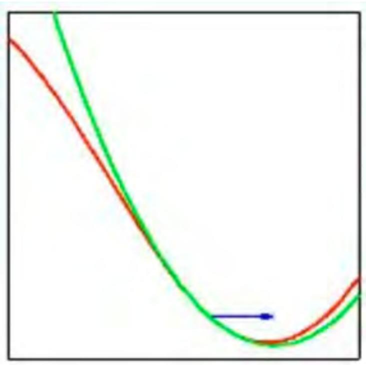
Newton‘s method

Δx_k minimizes a quadratic approximation of the nonlinear model

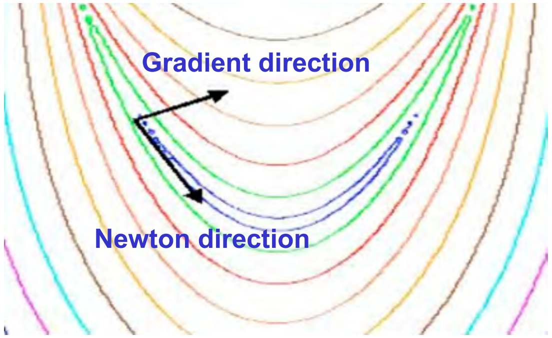
$$\boxed{Q(p^k) = f(x^k) + \nabla f(x^k) p^k + \frac{1}{2} p^{k^T} H^k p^k}$$

with $H^k = \nabla^2 f(x^k)$





If the quadratic model is a good approximation of the nonlinear model, then a full step can be performed ($\alpha_k = 1$), otherwise it has to be adapted



Convergence of Newton's method

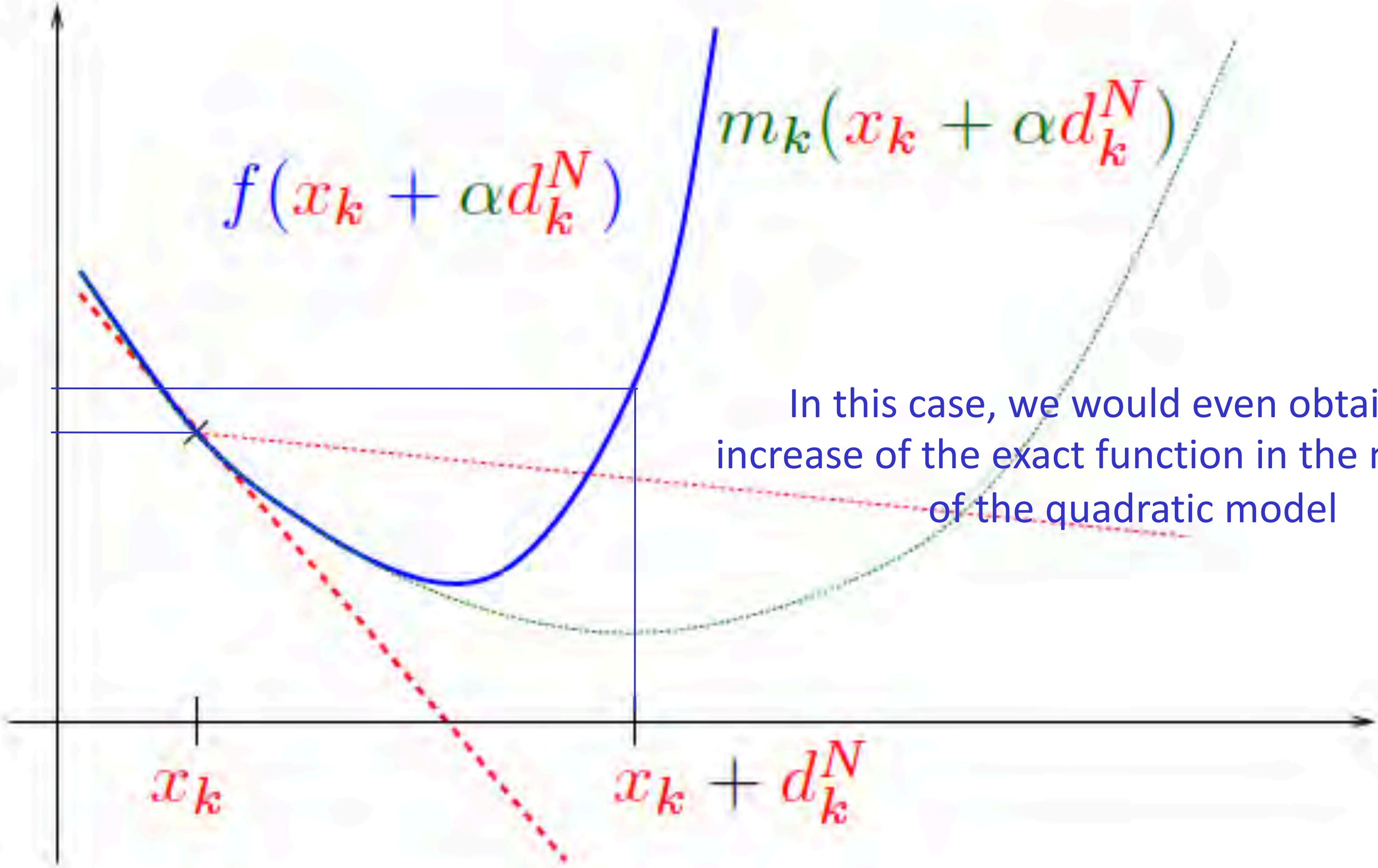
Newton's methods with full steps has **quadratic convergence**

i.e. $$\left\| x^k - x^* \right\| \leq C \left\| x^{k-1} - x^* \right\|^2$$

- **Problems:**
  - Convergence is only local (i.e. if start value is not too far away from the solution)
  - We would like to have "global convergence", i.e. convergence from every starting point.
  - Can be achieved by a good adjustment of step sizes (no full steps, also called damping of steps)
  - Quadratic convergence rate is lost.

Line search

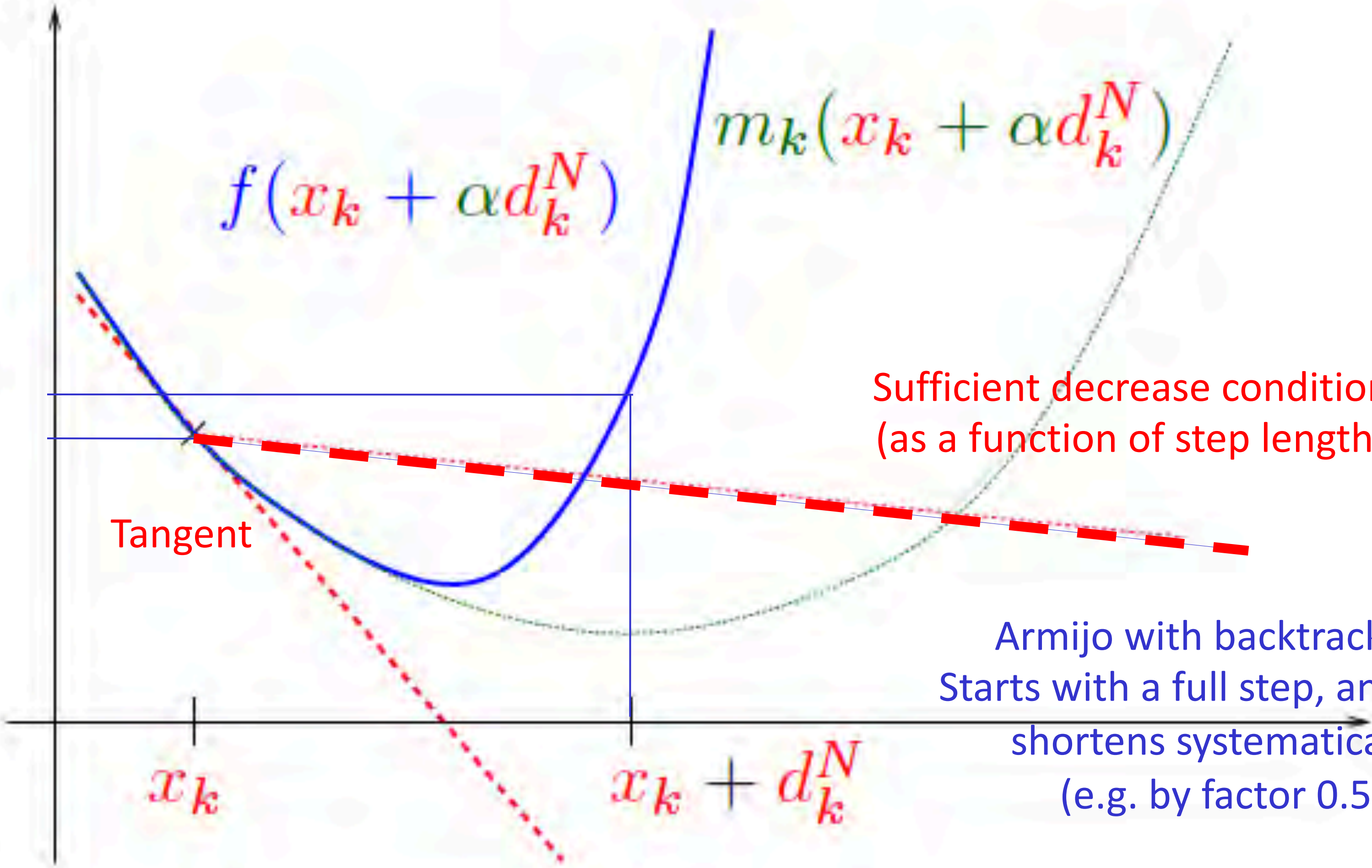
There is no need to look for the exact minimum along the ondimensional search direction, and often it even does not make sense. Instead a sufficient decrease is requested.



In this case, we would even obtain an increase of the exact function in the minimum of the quadratic model

Armijo Conditions for line search

$$f(x_k + \alpha_k p_k) \leq f(x_k) + c_1 \alpha_k \nabla f_k^T p_k \qquad c_1 \in (0,1)$$



Armijo with backtracking:
Starts with a full step, and then
shortens systematically
(e.g. by factor 0.5)

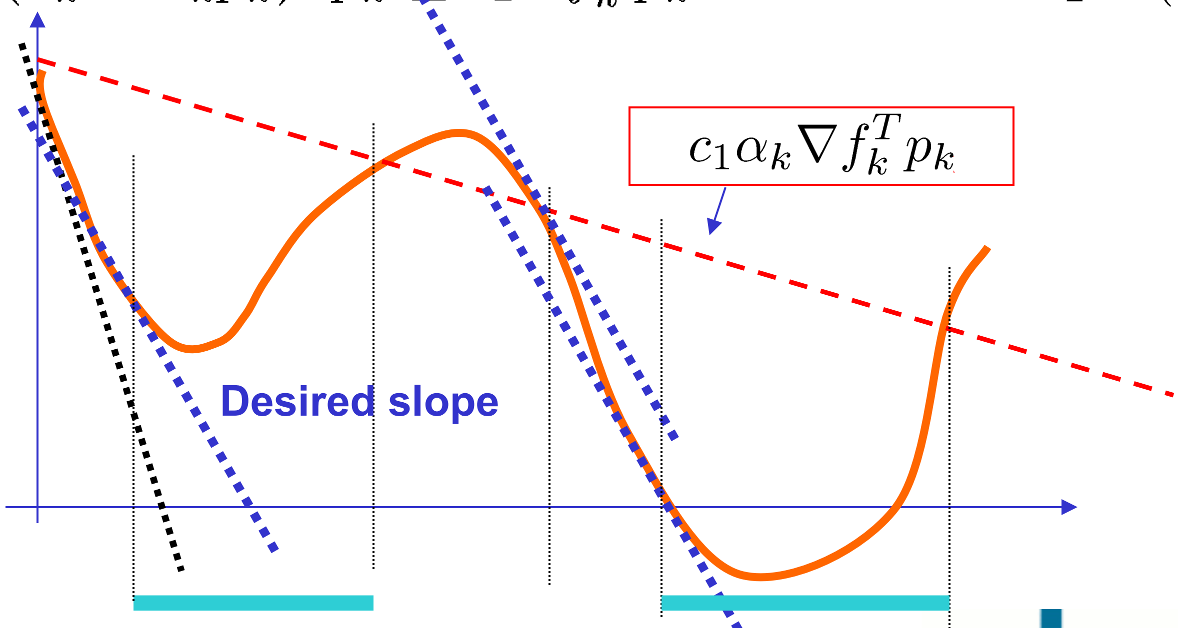
Wolfe conditions for line search

- Armijo condition (sufficent decrease):

$$f(x_k + \alpha_k p_k) \leq f(x_k) + c_1 \alpha_k \nabla f_k^T p_k \qquad c_1 \in (0,1)$$

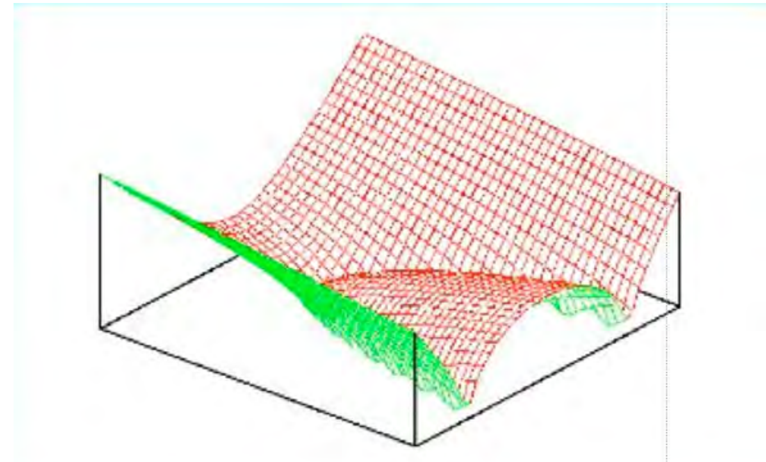
- Curvature condition

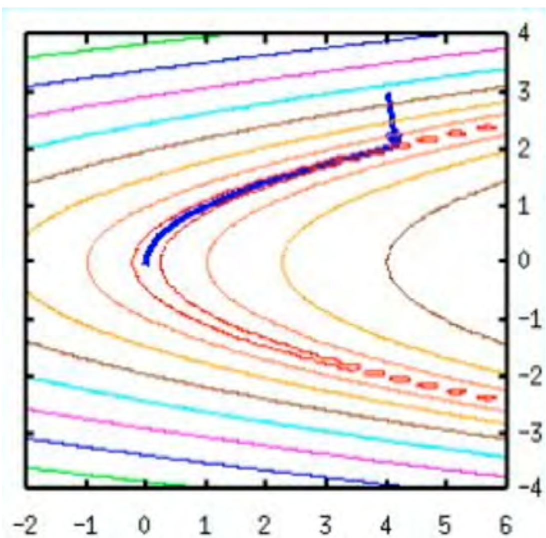
$$\nabla f(x_k + \alpha_k p_k)^T p_k \geq c_2 \nabla f_k^T p_k \qquad c_2 \in (c_1, 1)$$

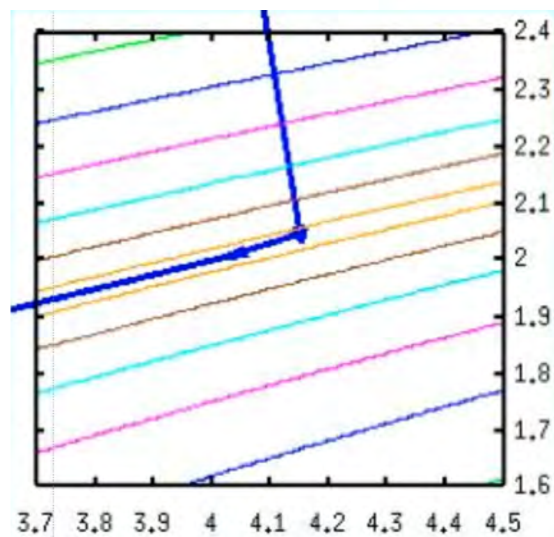


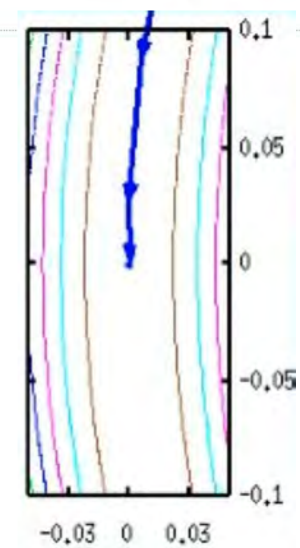
Example – Newton's method

- Behavior for banana valley function



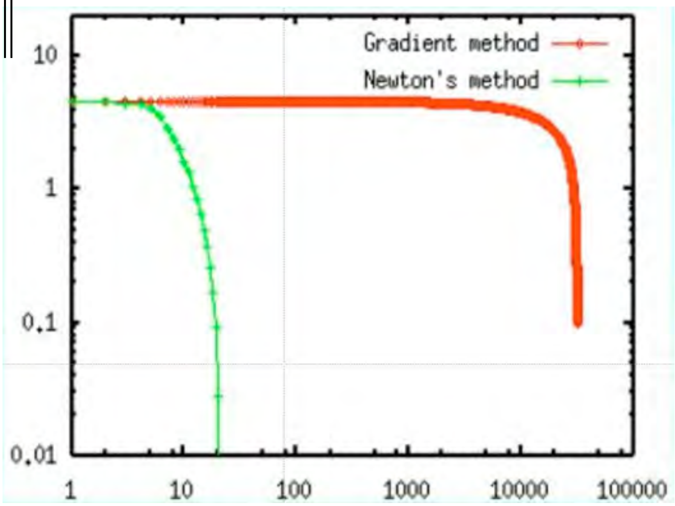






Banana valley example: Comparison between the two methods

$\left\|x^k - x^*\right\|$



- Newton‘s method is much faster than steepest descent
- Convergence is nearly linear for the first 10 iterations since setp length

$$\alpha^k < 1$$

- Convergence is roughly quadratic for the last iterations with

$$\alpha^k \approx 1$$


Thank you very much for your attention!