

Nonlinear Optimization and Direct Optimal Control for Practitioners

Moritz Diehl

Systems Control and Optimization Laboratory

Department of Microsystems Engineering and Department of Mathematics
University of Freiburg

joint work with Katrin Baumgärtner and Florian Messerer

SPP2364 Online Seminar,
June 26, 2023

What to do with models ?

- Simulation
- Optimization (offline, human-in-the-loop)
- **Embedded Optimization (online, without human interaction)**

“Optimization friendly” models (differentiability, convexity, ...) allow one to use reliable and fast optimization algorithms

Complex Sensor Actuator Systems

SENSORS

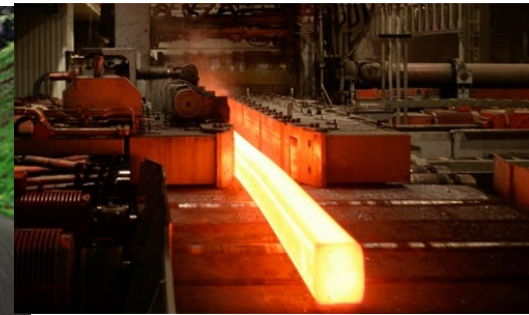
- GPS
- acceleration
- radar
- vision
- ...



How to connect ?

ACTUATORS

- flight surfaces
- steering wheel
- motor speeds
- torques
- ...



Complex Sensor Actuator Systems

SENSORS

- GPS
- acceleration
- radar
- vision
- ...



How to connect ?
Linear Filters ?

ACTUATORS

- flight surfaces
- steering wheel
- motor speeds
- torques
- ...



Complex Sensor Actuator Systems

SENSORS

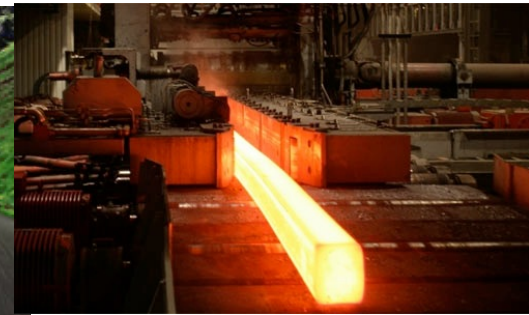
- GPS
- acceleration
- radar
- vision
- ...



How to connect ?
Linear Filters ?
Deep Neural Networks ?

ACTUATORS

- flight surfaces
- steering wheel
- motor speeds
- torques
- ...



Complex Sensor Actuator Systems

SENSORS

- GPS
- acceleration
- radar
- vision
- ...

EMBEDDED OPTIMIZATION

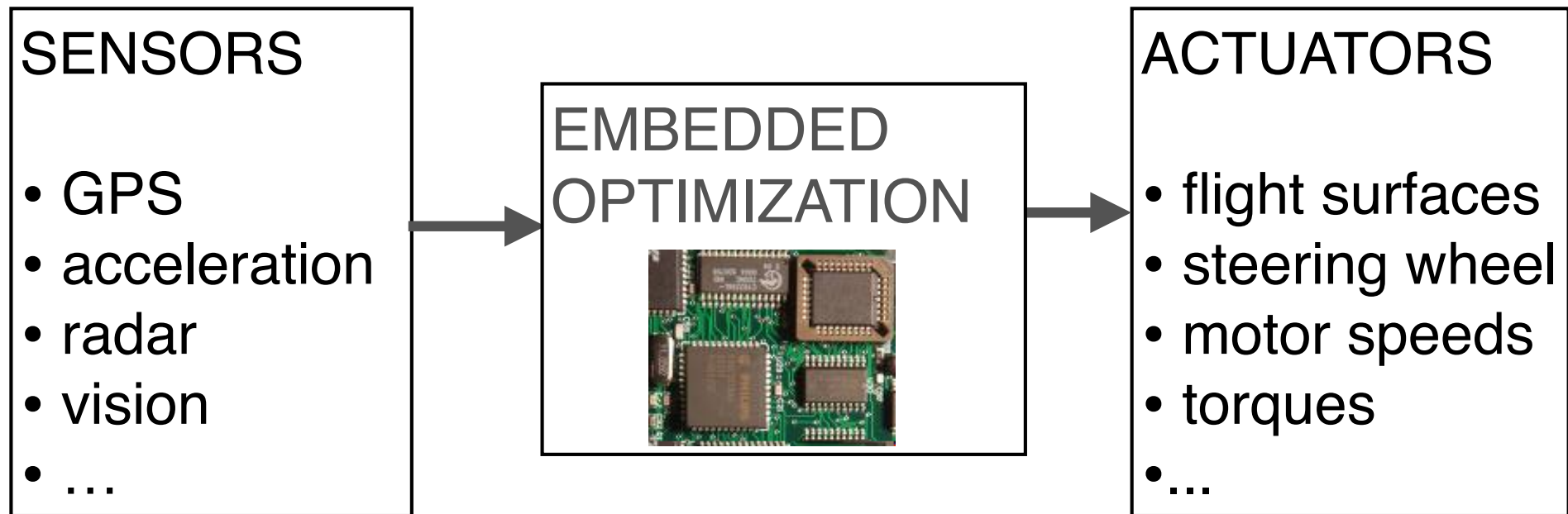


ACTUATORS

- flight surfaces
- steering wheel
- motor speeds
- torques
- ...



Complex Sensor Actuator Systems

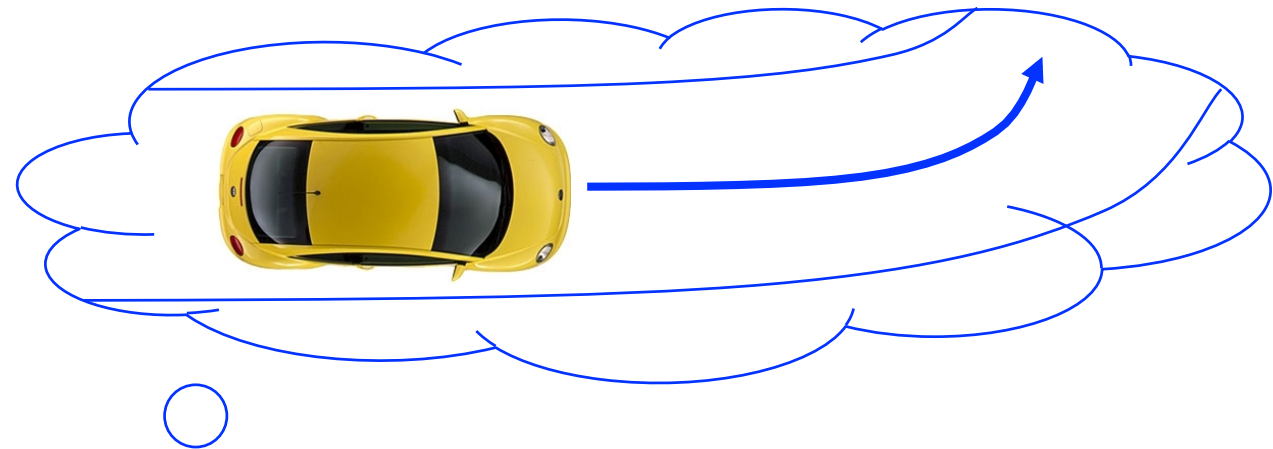


Solve, in real-time and repeatedly, an optimization problem that depends on the incoming stream of input data, to generate a stream of output data.

Embedded Optimization: a CPU intensive map

Embedded Optimization for Model Predictive Control (MPC)

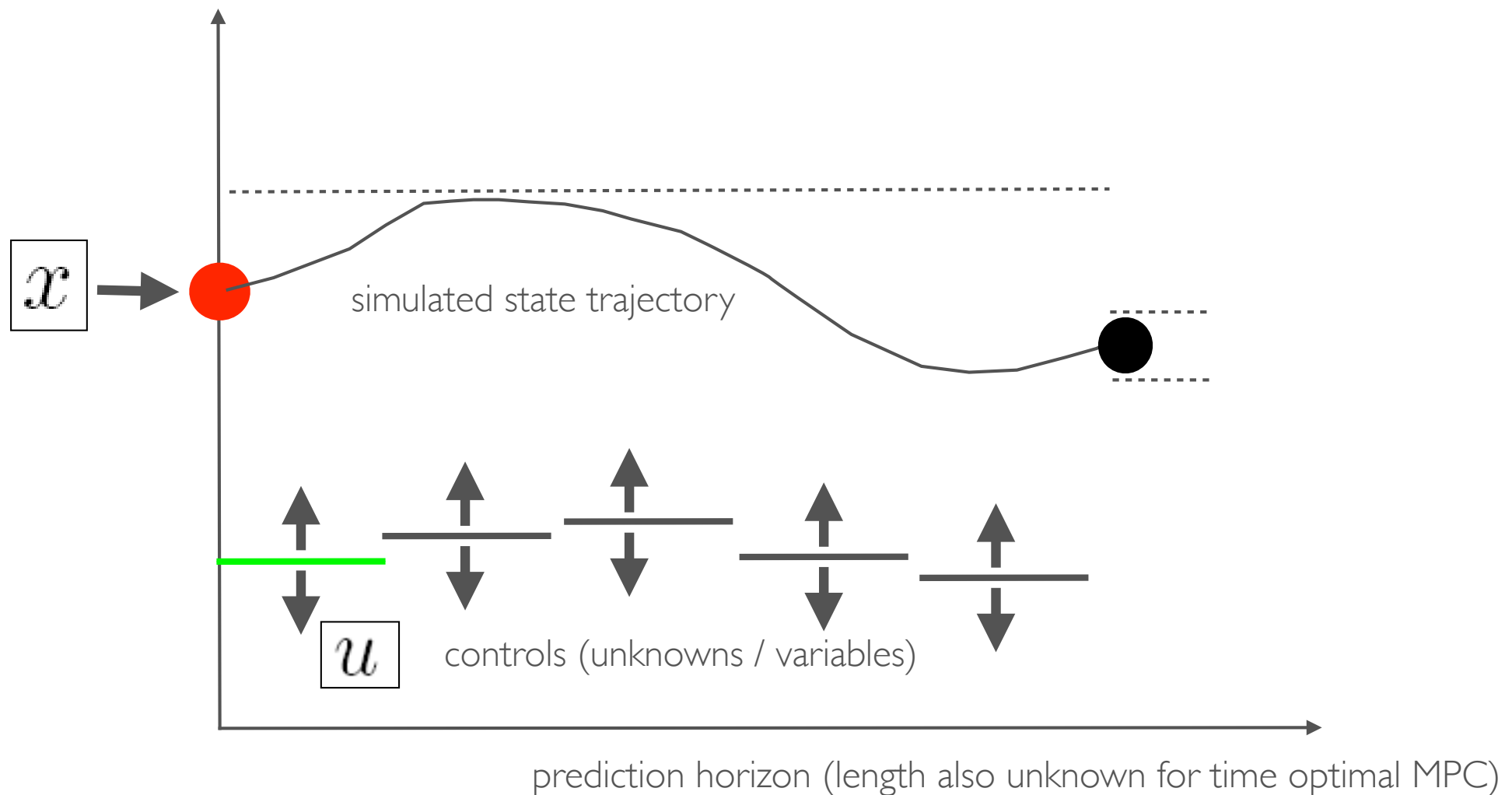
Always look a bit into the future



Example: driver predicts and optimizes, and therefore slows down before a curve

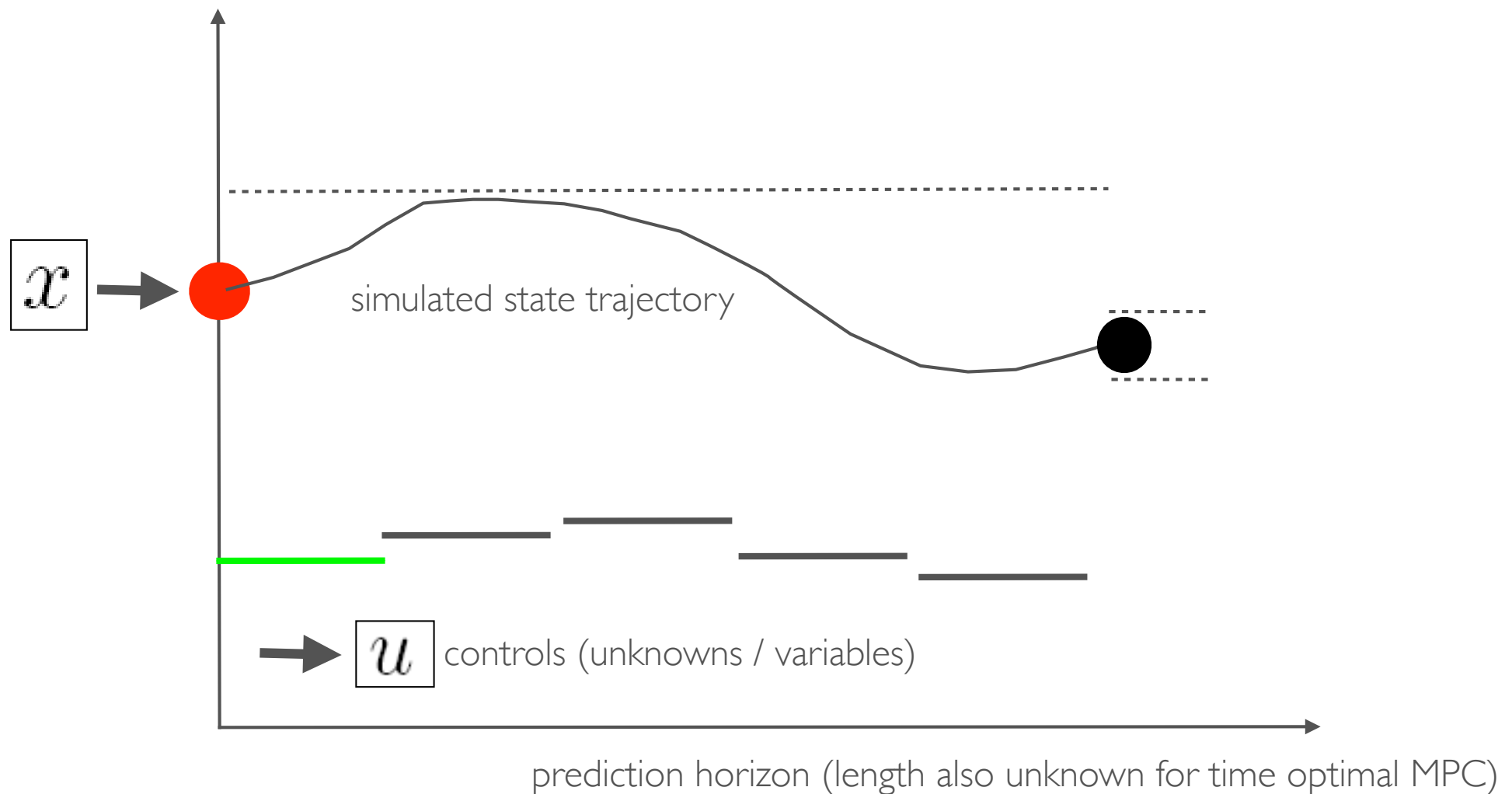
Optimal Control Problem in MPC

For given system state \mathbf{x} , which controls \mathbf{u} lead to the best objective value without violation of constraints ?

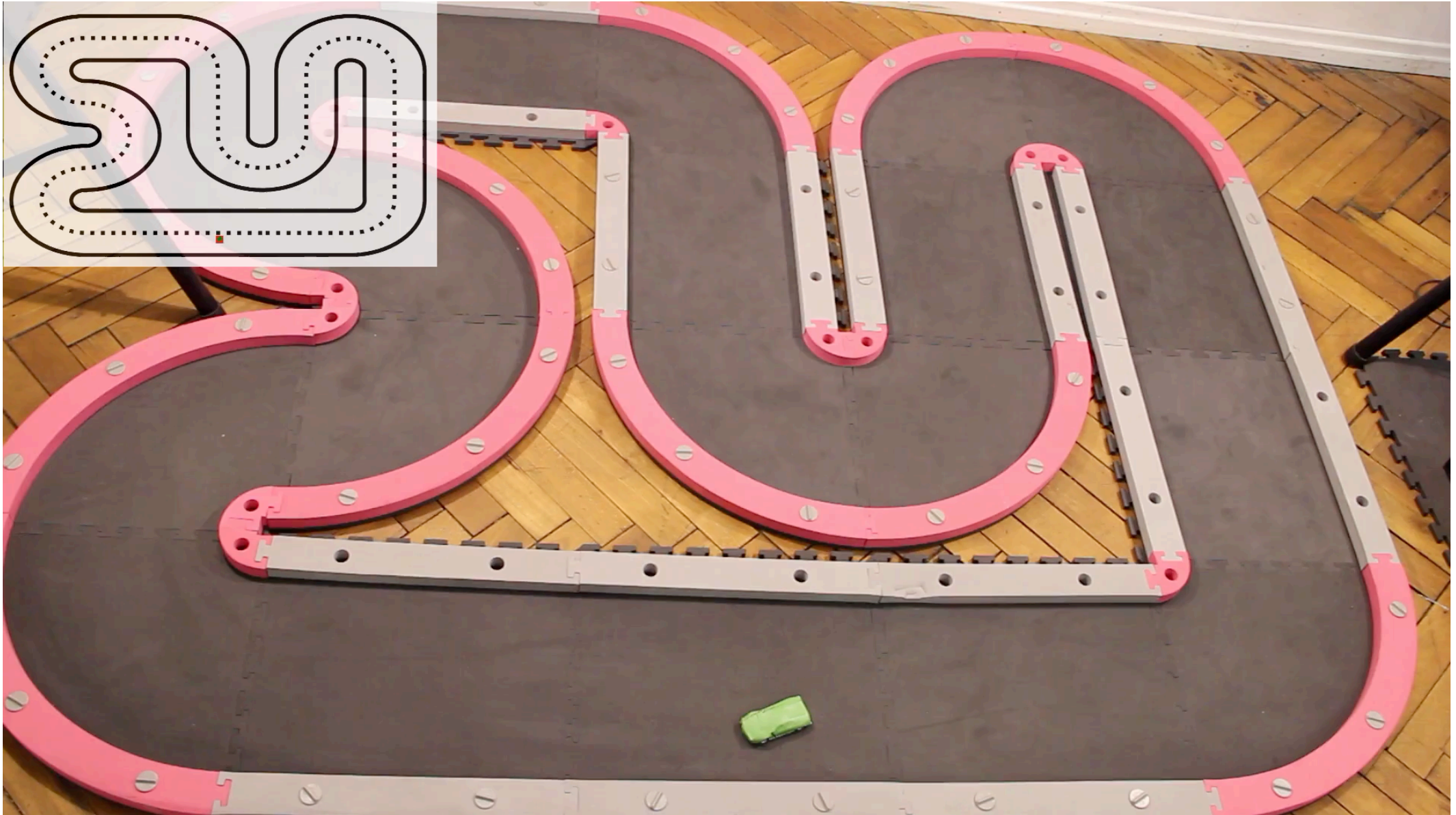


Optimal Control Problem in MPC

For given system state \mathbf{x} , which controls \mathbf{u} lead to the best objective value without violation of constraints ?



Model Predictive Control of the Freiburg Race Cars



acados coupled into ROS, optimization every 10ms

[Kloeser et al., 2020]

2nd MPC Example: Time-Optimal Point-To-Point Motions



Fast oscillating systems (cranes, plotters, wafer steppers, ...)

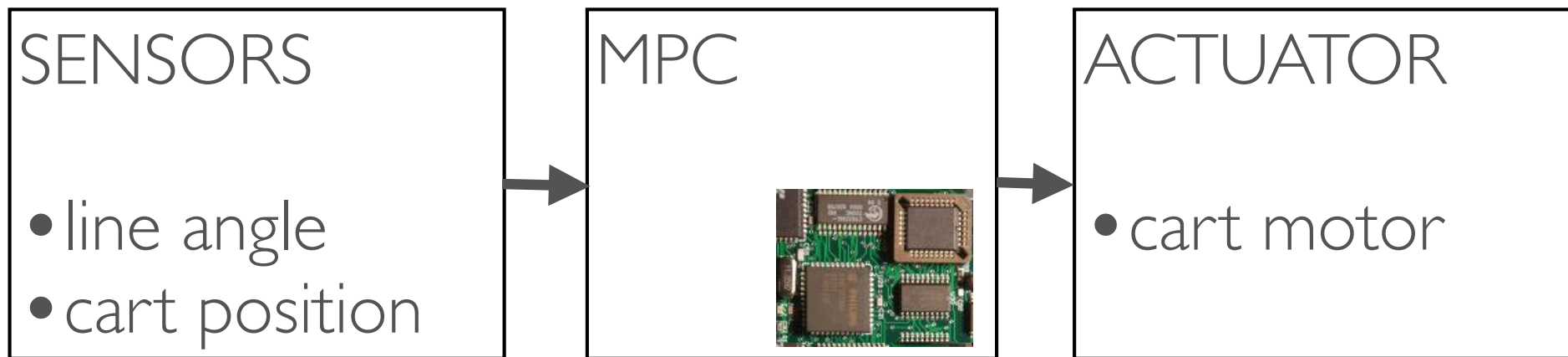
Control aims:

- reach end point as fast as possible
- do not violate constraints
- no residual vibrations

Idea: formulate as embedded optimization problem
in form of Model Predictive Control (MPC)



Time Optimal MPC of a Crane

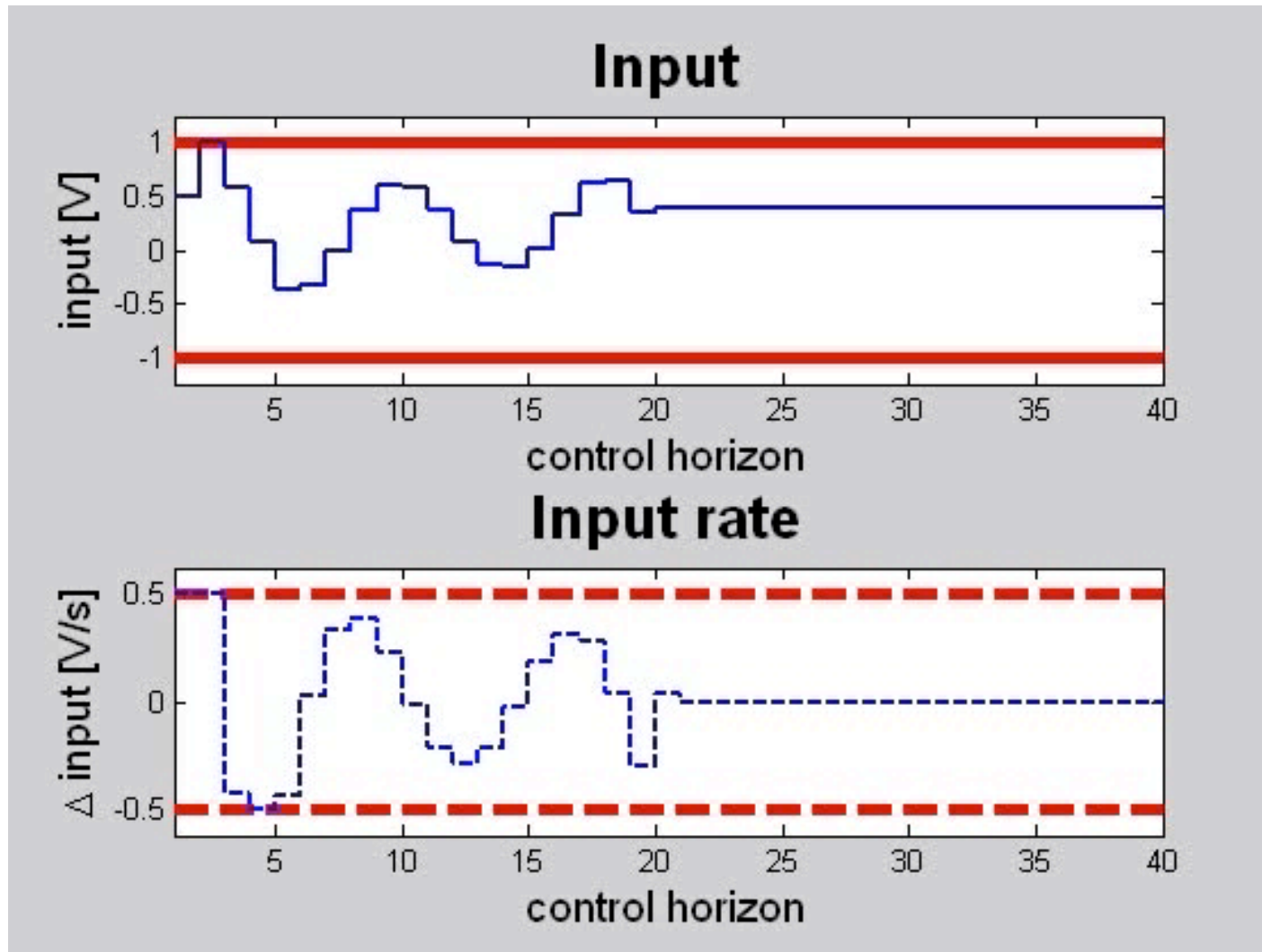


Time Optimal MPC of a Crane

Univ. Leuven [Vandenbrouck, Swevers, D.]

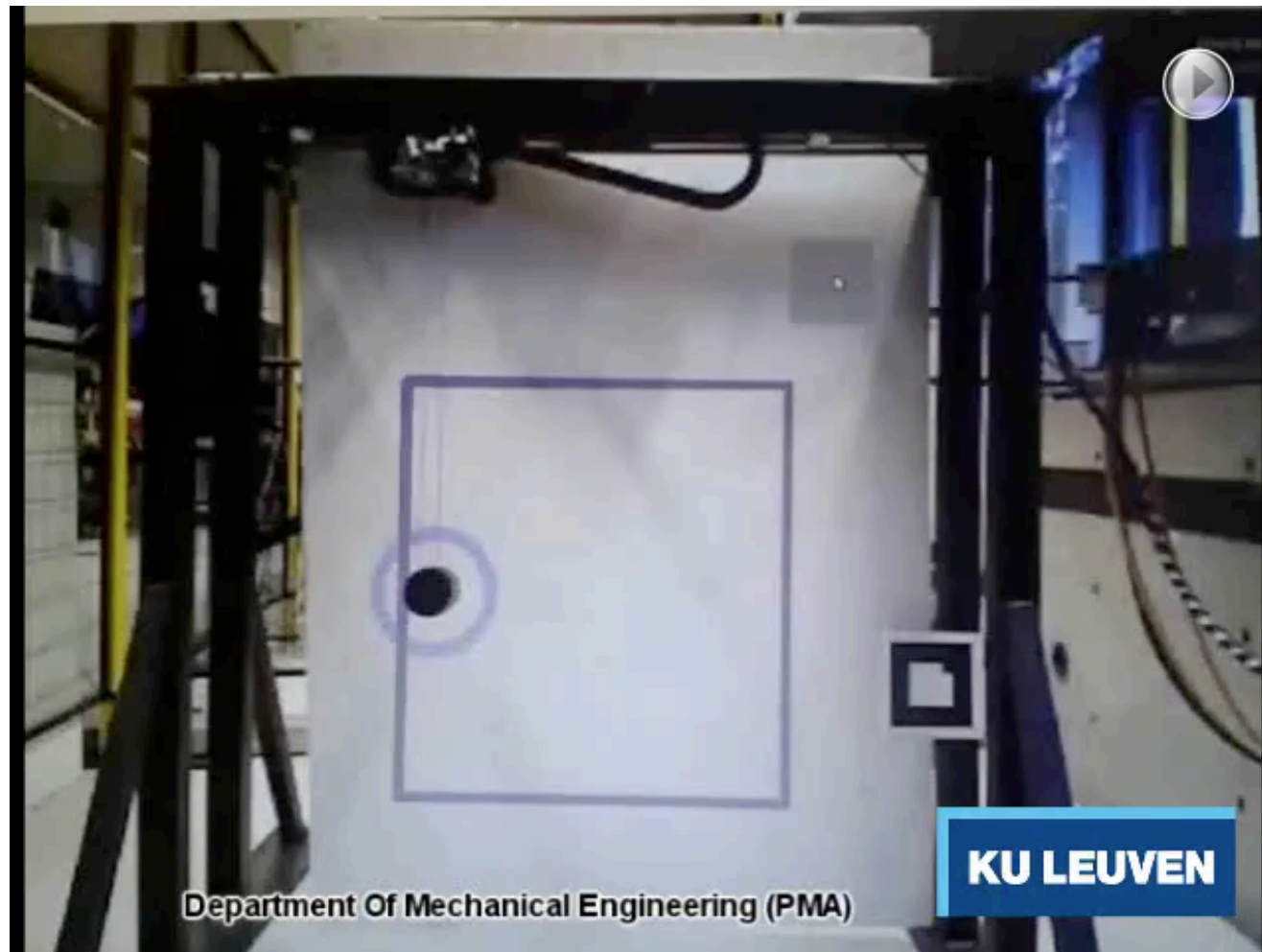


Optimal Solutions in qpOASES Varying in Time



Time Optimal “drawing” by crane

Univ. Leuven [Wannes Van Loock et al.,] (CasADi)



Conclusion

Nonlinear optimization can be very fast and reliable with optimisation-friendly models (and good algorithms)

Nonlinear Optimization and Direct Optimal Control for Practitioners

Katrin Baumgärtner, Florian Messerer, Prof. Dr. Moritz Diehl

General information

This online workshop consists of two parts:

- 26.06.2023, 12:00-13:30 – Part 1: Nonlinear Optimization
- 03.07.2023, 12:00-13:30 – Part 2: Direct Optimal Control and Model Predictive Control

The aim of this workshop is to give you some hands on experience on methods and in particular software for optimal control. The workshop exercises are based on `python`, `CasADi`, and `acados`.



<http://casadi.org>

- A software framework for nonlinear optimization and optimal control
- “Write an efficient optimal control solver in a few lines”
- Implements automatic differentiation (AD) on sparse matrix-valued computational graphs in C++11
- Front-ends to Python, Matlab and Octave
- Supports C code generation
- Back-ends to SUNDIALS, CPLEX, Gurobi, qpOASES, IPOPT, KNITRO, SNOPT, SuperSCS, OSQP, ...
- Developed by Joel Andersson and Joris Gillis



[Andersson, Gillis, Horn, Rawlings, D., Math. Prog. Comp., 2019]

Today's Toy Problem for Nonlinear Optimization

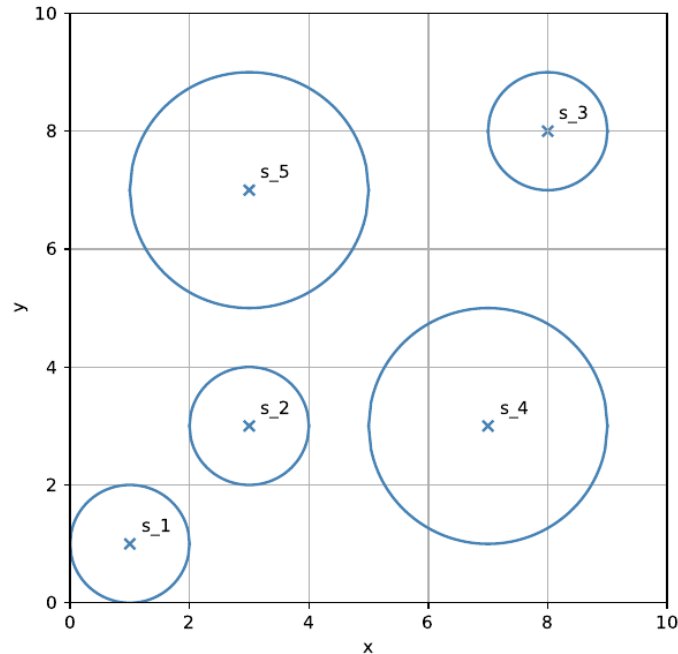


Figure 1: Graphical depiction of a possible, but suboptimal solution with $R = 1$.

The problem can be formulated as a nonlinear program in `CasADi` and solved using `IPOPT`, where the

Optimization: an Overview

Moritz Diehl
University of Freiburg

(some slide material was provided by W. Bangerth and K. Mombaur)

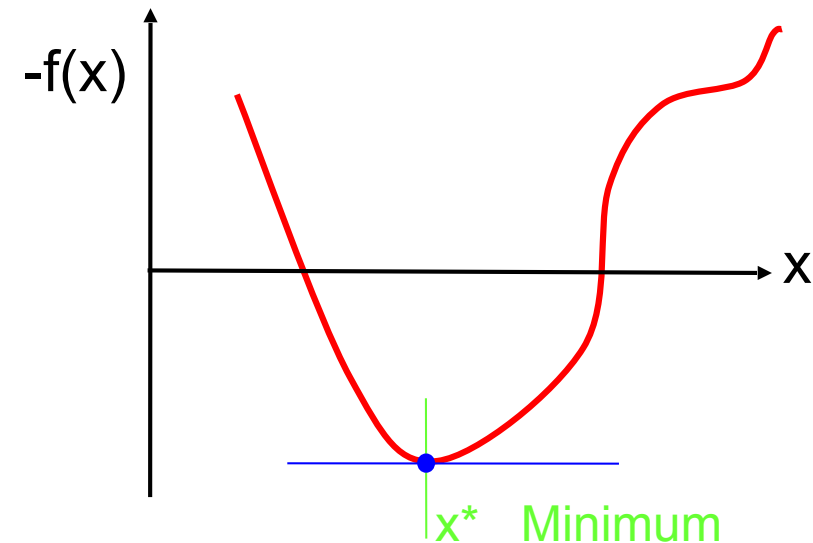
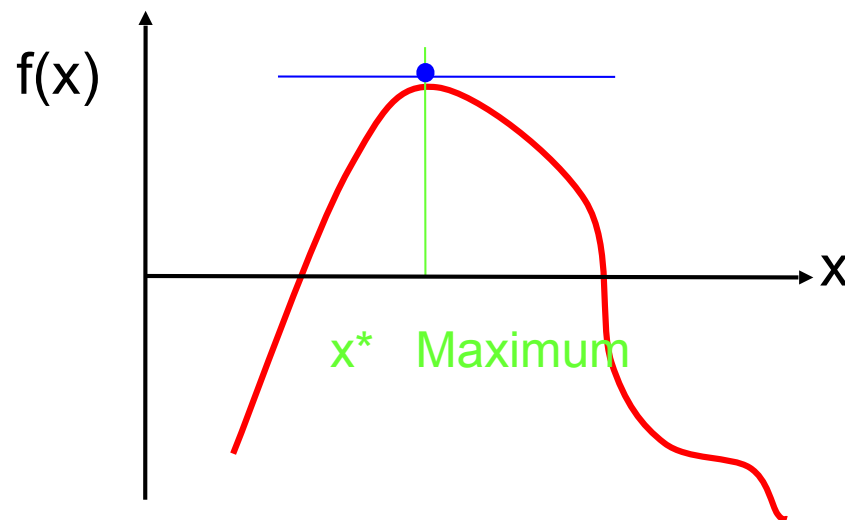
Overview of presentation

- **Optimization: basic definitions and concepts**
- Introduction to classes of optimization problems

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 maximization and minimization problems:
(from now on only minimization)



Constrained optimization

- Often variable x shall satisfy certain constraints, e.g.:
 - $x \geq 0$
 - $x_1^2 + x_2^2 = C$
- General formulation:

$$\min f(x)$$

subject to (s.t.)

$$g(x) = 0$$

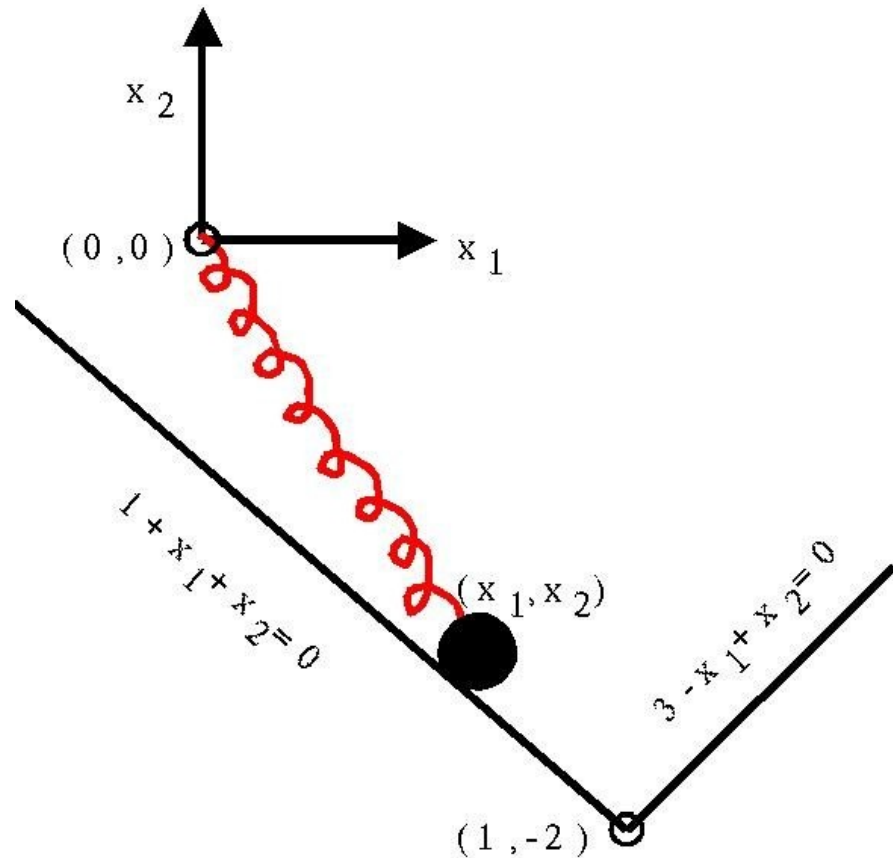
$$h(x) \geq 0$$

f objective function / cost function

g equality constraints

h inequality constraints

Simple example: Ball hanging on a spring



To find position at rest,
minimize potential energy!

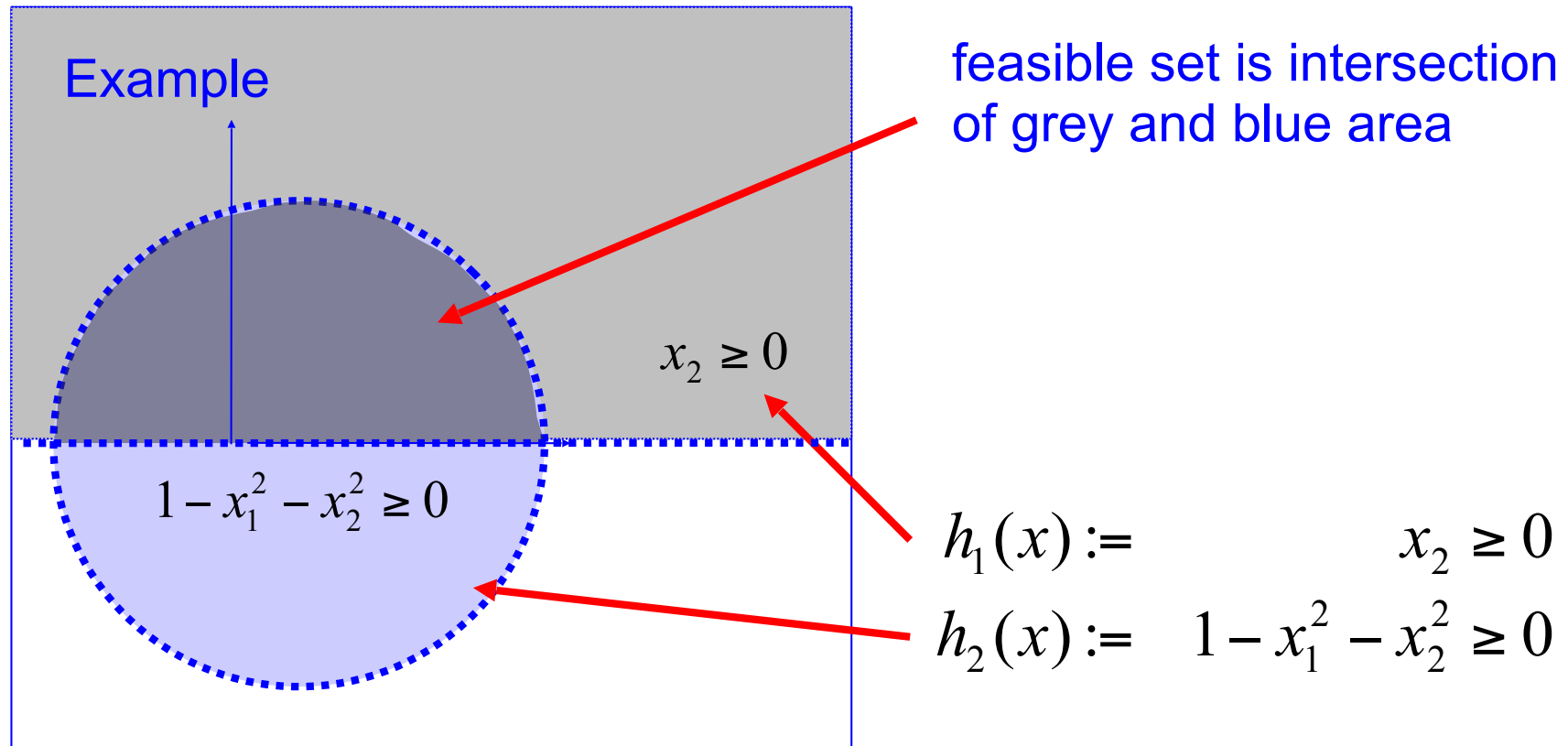
$$\min \underbrace{x_1^2 + x_2^2}_{\text{spring}} + \underbrace{mx_2}_{\text{gravity}}$$

$$1 + x_1 + x_2 \geq 0$$

$$3 - x_1 + x_2 \geq 0$$

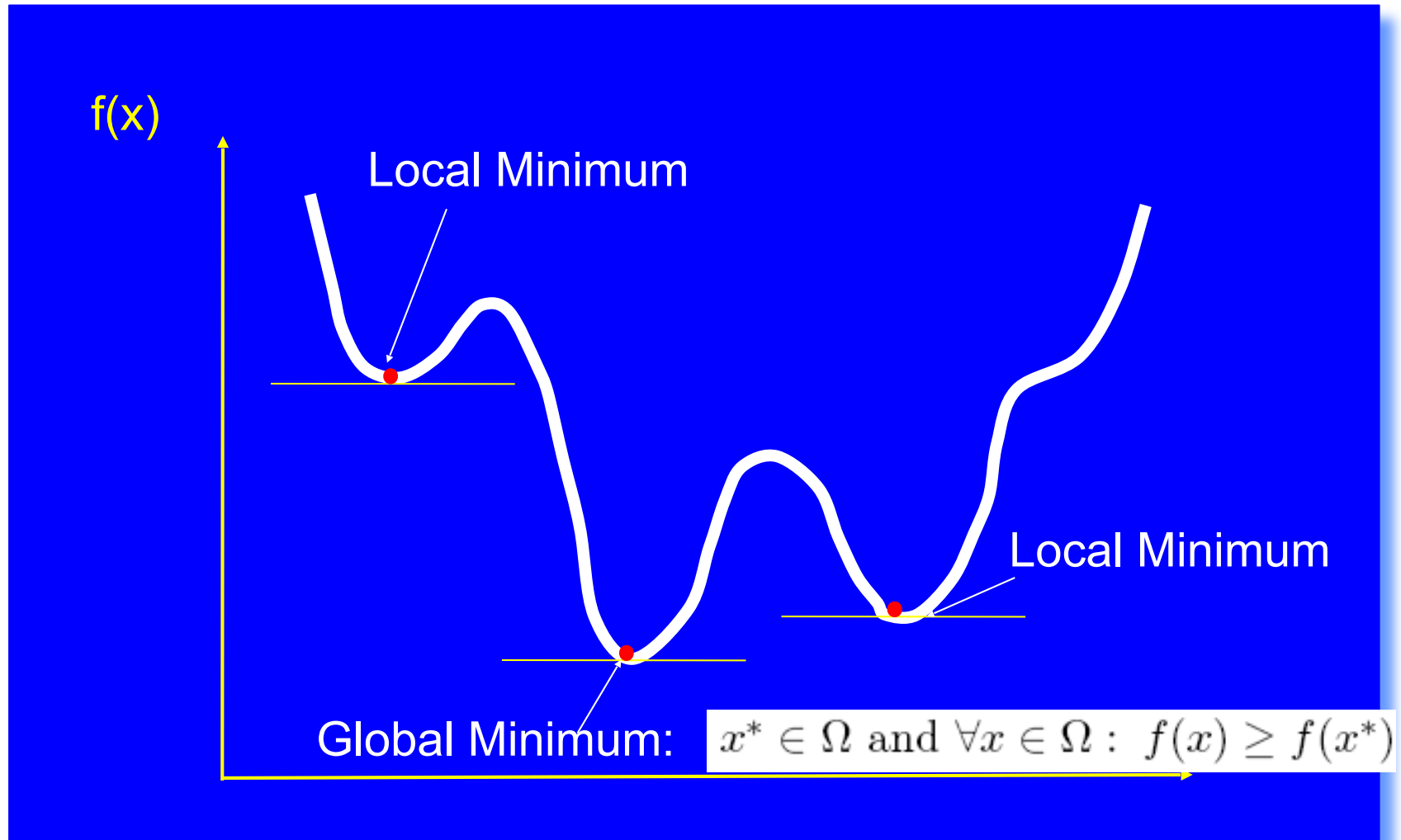
Feasible set

Feasible set = collection of all points that satisfy all constraints:



The “feasible set” Ω is $\{x \in \mathbb{R}^n \mid g(x) = 0, h(x) \geq 0\}$.

Local and global optima



The point $x^* \in \mathbb{R}^n$ is a “local minimizer” iff $x^* \in \Omega$ and there exists a neighborhood \mathcal{N} of x^* (e.g. an open ball around x^*) so that $\forall x \in \Omega \cap \mathcal{N} : f(x) \geq f(x^*)$.

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)

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

- and the second order derivatives are called the **Hessian matrix**

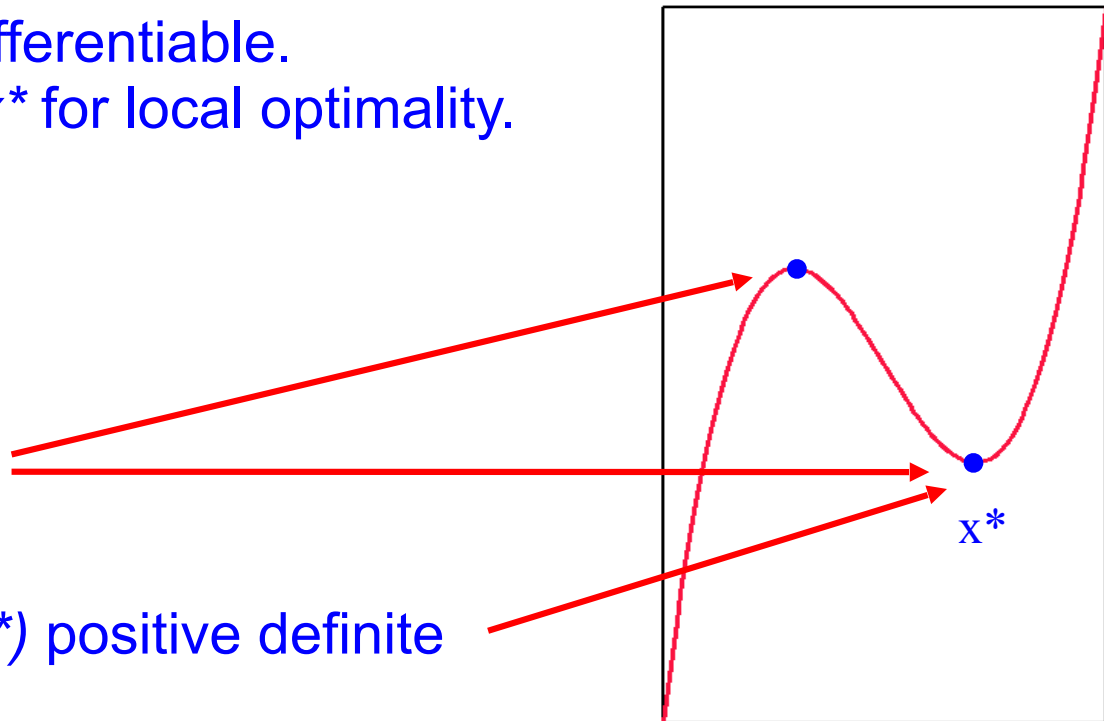
$$\nabla^2 f(x) = \begin{pmatrix} \frac{\partial^2 f}{\partial x_1^2} & \frac{\partial^2 f}{\partial x_1 \partial x_2} & \cdots & \frac{\partial^2 f}{\partial x_1 \partial x_n} \\ \frac{\partial^2 f}{\partial x_2 \partial x_1} & \frac{\partial^2 f}{\partial x_2^2} & \cdots & \frac{\partial^2 f}{\partial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

Optimality conditions (unconstrained)

$$\min f(x) \quad x \in \mathbb{R}^n$$

Assume that f is twice differentiable.
We want to test a point x^* for local optimality.

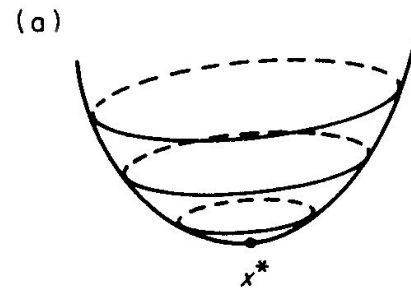
- *necessary condition:*
 $\nabla f(x^*) = 0$ (stationarity)
- *sufficient condition:*
 x^* stationary and $\nabla^2 f(x^*)$ positive definite



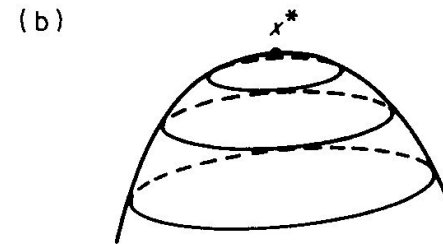
Types of stationary points

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

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

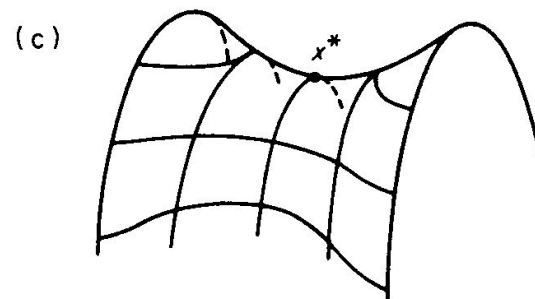


Minimum



Maximum

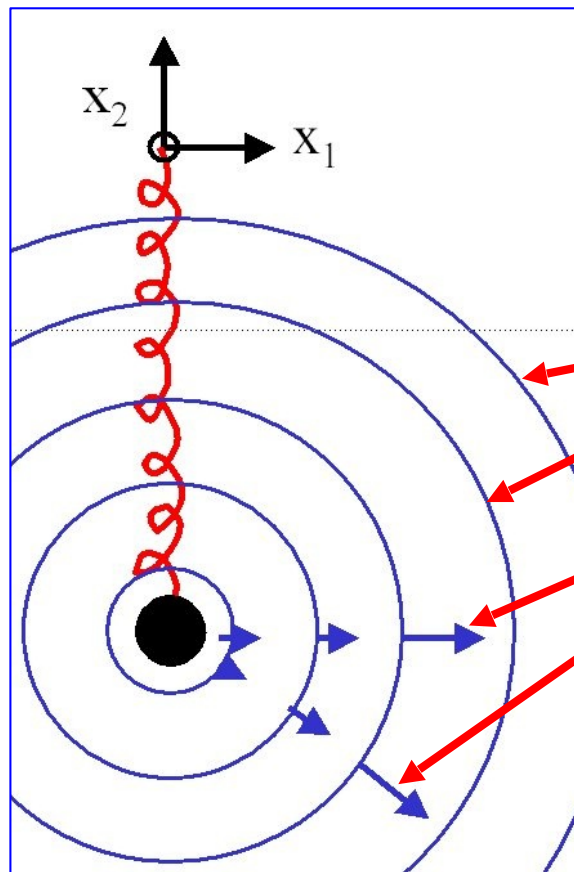
$\nabla^2 f(x^*)$ negative definite:
local maximum



Saddle

$\nabla^2 f(x^*)$ indefinite: saddle point

Ball on a spring without constraints



$$\min_{x \in \mathbb{R}^2} x_1^2 + x_2^2 + mx_2$$

contour lines of $f(x)$

gradient vector

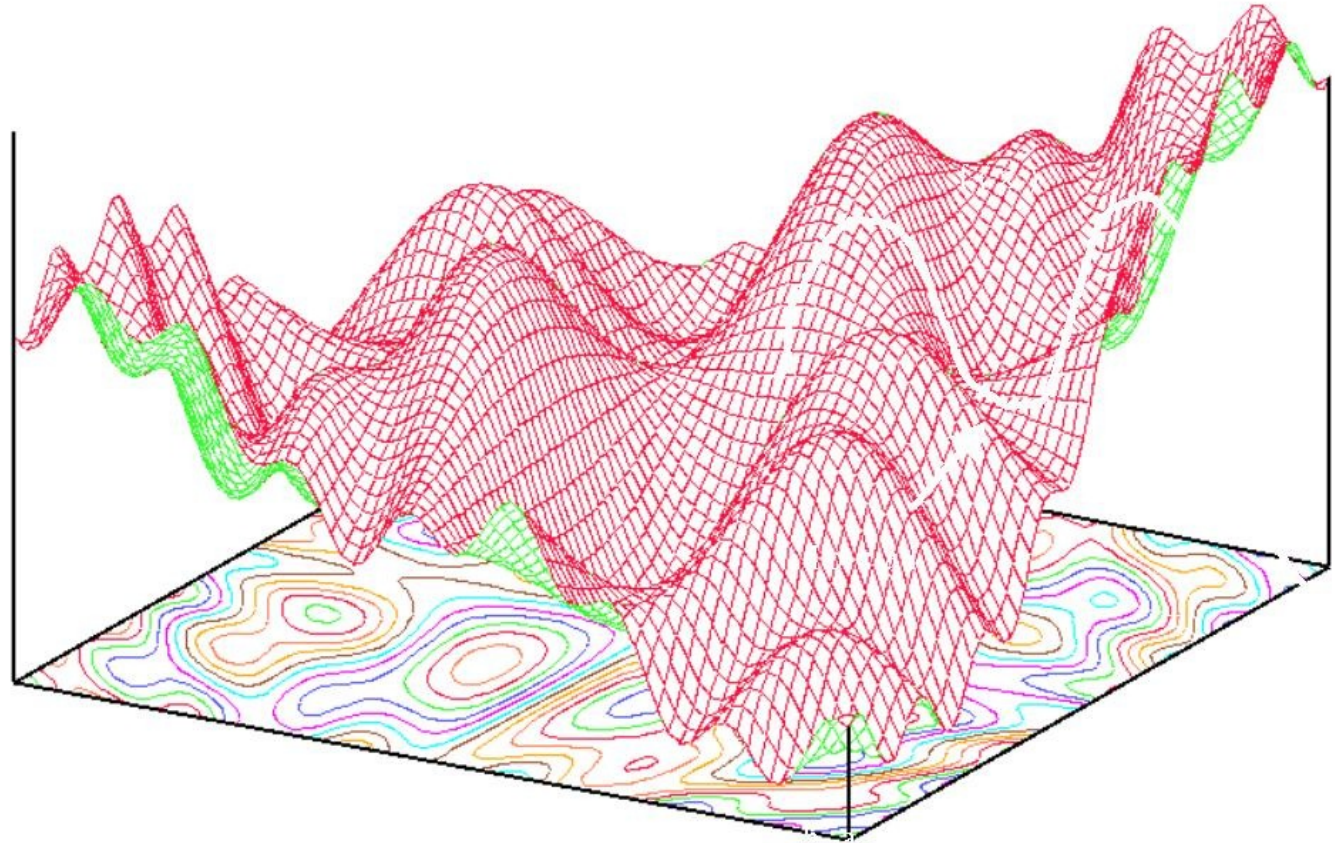
$$\nabla f(x) = (2x_1, 2x_2 + m)$$

unconstrained minimum:

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

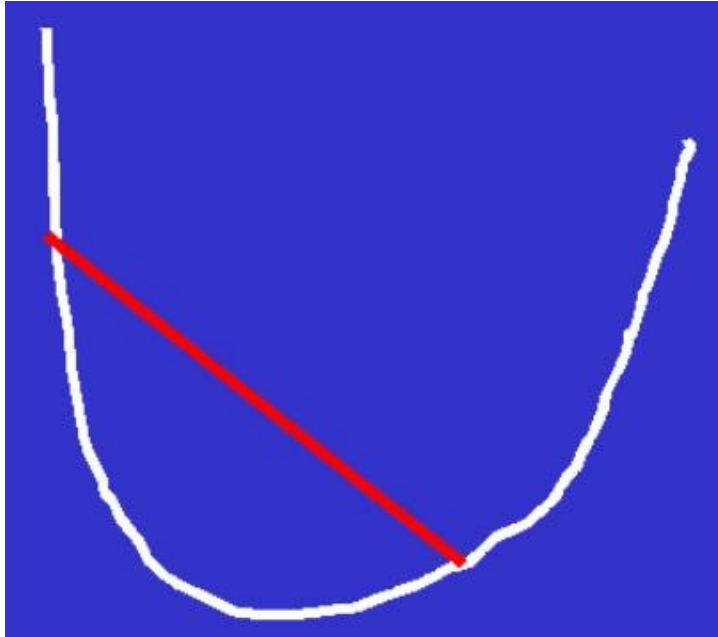
Sometimes there are many local minima

e.g. potential energy of macromolecule

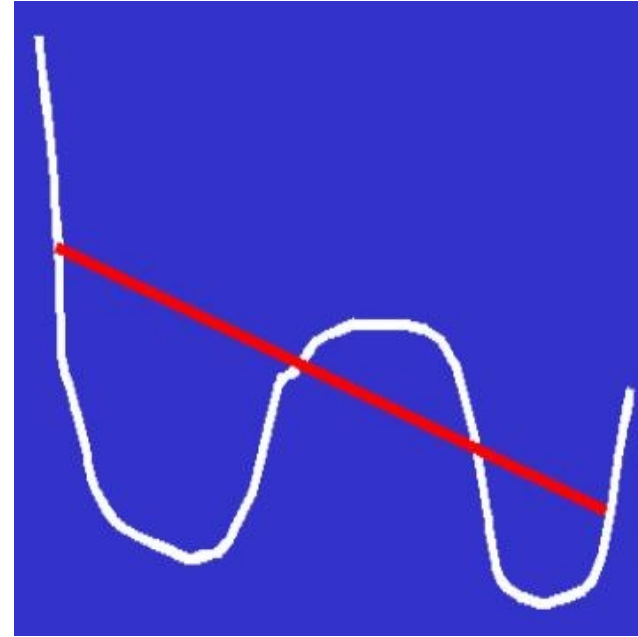


Global optimization is a very hard issue - most algorithms find only the next local minimum. But there is a favourable special case...

Convex functions

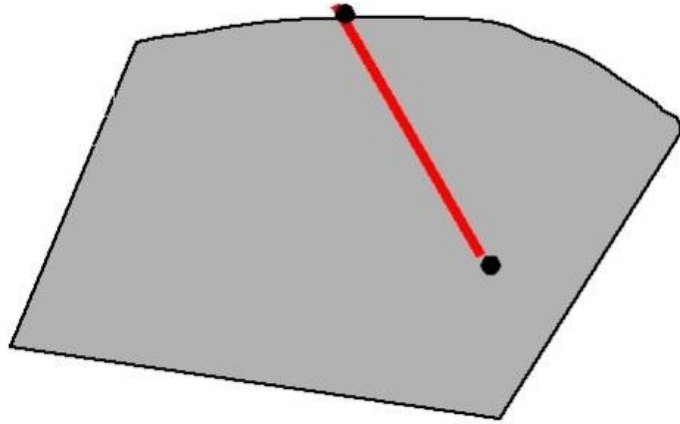


Convex: all connecting lines are above graph

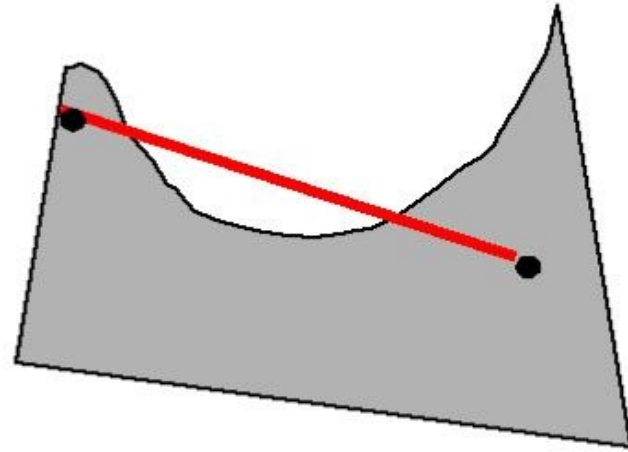


Non-convex: some connecting lines are not above graph

Convex feasible sets

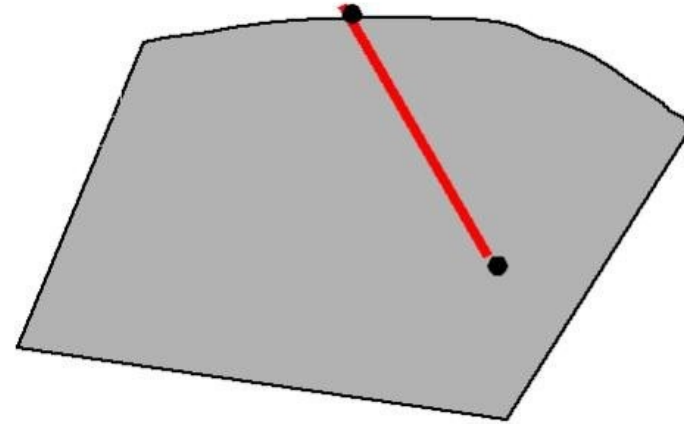
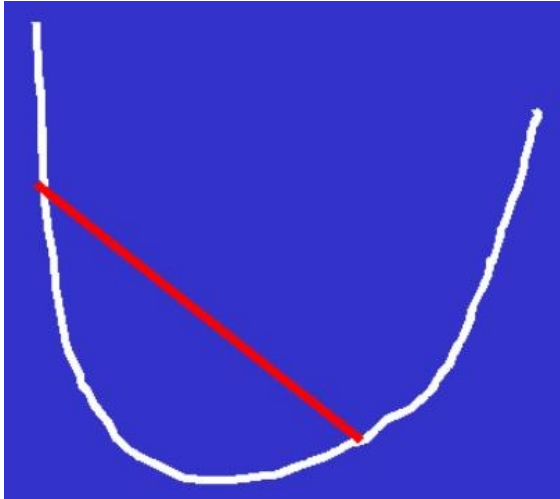


Convex: all connecting lines between feasible points are in the feasible set



Non-convex: some connecting line between two feasible points is not in the feasible set

Convex problems



Convex problem if

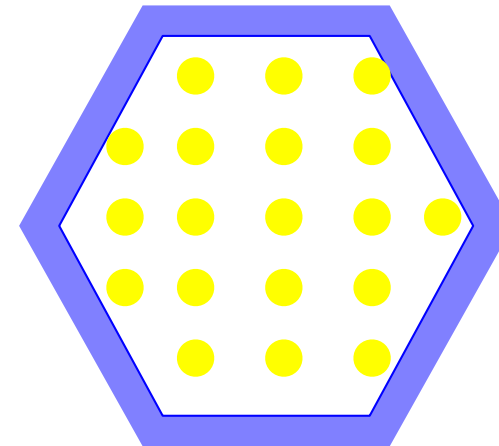
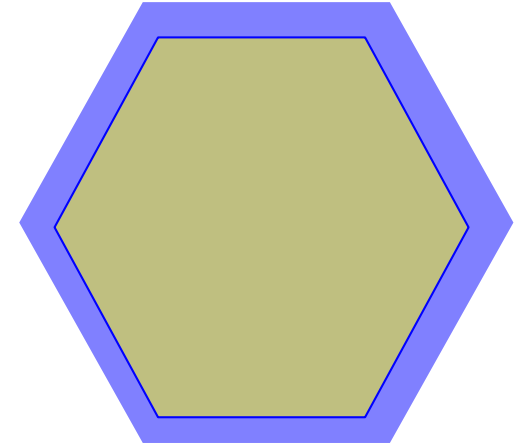
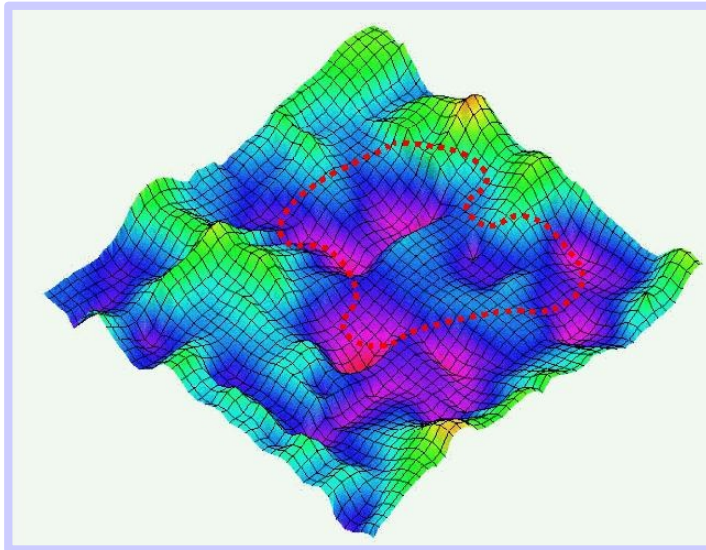
$f(x)$ is convex and the feasible set is convex

One can show:

**For convex problems, every local minimum is also a global minimum.
It is sufficient to find local minima!**

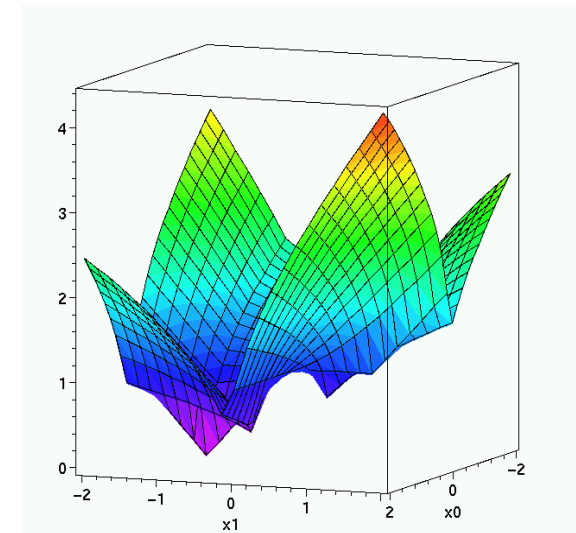
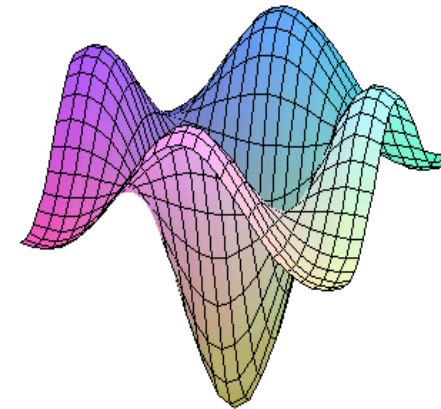
Characteristics of optimization problems 1

- 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, dynamic equations (optimal control)
 - smoothness



Overview of presentation

- Optimization: basic definitions and concepts
- **Introduction to classes of optimization problems**


Problem Class 1: Linear Programming (LP)

- Linear objective,
linear constraints:
Linear Optimization Problem
(convex)

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

- 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

Problem Class 2: Quadratic Programming (QP)

- Quadratic objective and linear constraints:
Quadratic Optimization Problem
(convex, if Q pos. def.)

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

- 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**

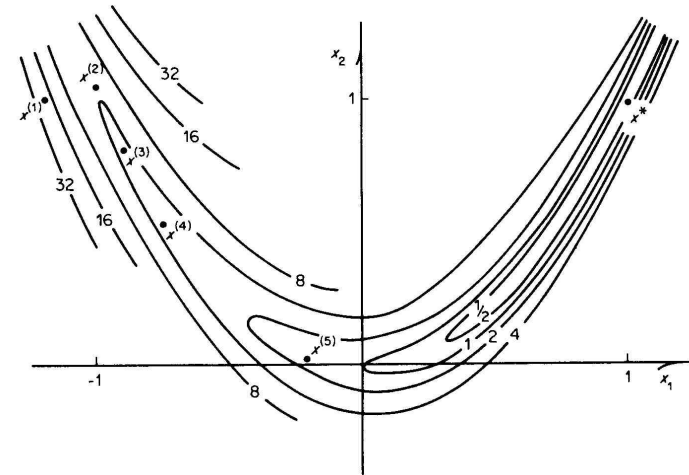
Problem Class 3: Nonlinear Programming (NLP)

- Nonlinear Optimization Problem
(in general nonconvex)

$$\begin{array}{ll} \min_x & f(x) \\ \text{s. t.} & h(x) = 0 \\ & g(x) \geq 0 \end{array}$$

- E.g. the famous nonlinear Rosenbrock function

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



Problem Class 4: Non-smooth optimization

- objective function or constraints are non-differentiable or not continuous e.g.

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

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

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

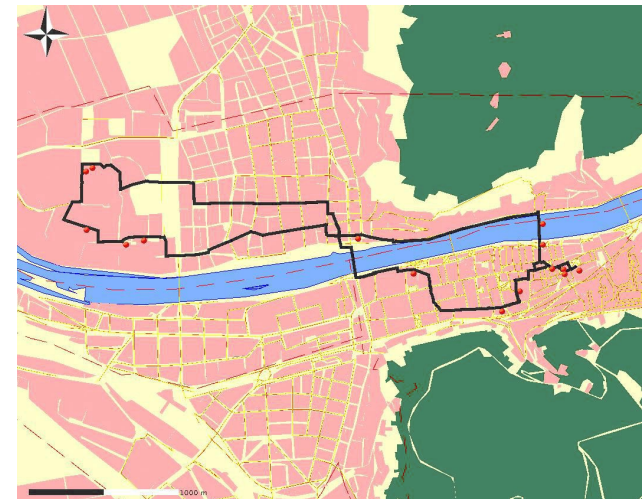
$$f(x) = i \quad \text{for} \quad i \leq x < i + 1, \quad i = 0, 1, 2, \dots$$

Problem Class 5: Integer Programming (IP)

- Some or all variables are integer (e.g. linear integer problems)

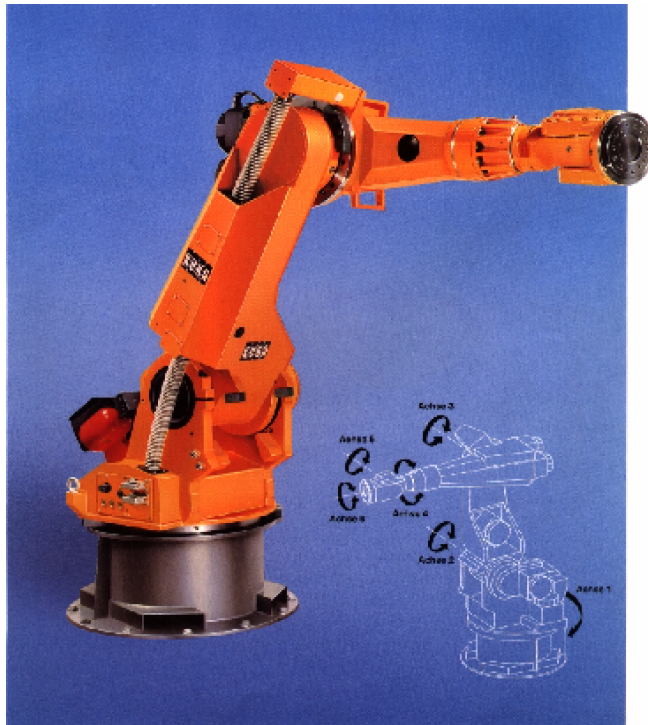
$$\begin{array}{ll} \min_x & c^T x \\ \text{s. t.} & Ax = b \\ & x \in Z_+^n \end{array}$$

- Special case: combinatorial optimization problems -- feasible set is finite
- Example: traveling salesman problem
 - determine fastest/shortest round trip through n locations



Problem Class 6: Optimal Control

- Optimization problems including dynamics in form of **differential equations** (infinite dimensional)



Variables $x(t), u(t), p$ (partly ∞ -dim.)

$$\min_{x,u,p} \int_0^T \phi(t, x(t), u(t), p) dt$$

$$\text{s. t. } \dot{x} = f(t, x(t), u(t), p)$$

....

THIS COURSE'S MAIN TOPIC!

Summary: Optimization Overview

Optimization problems can be:

- unconstrained or constrained
- convex or non-convex
- linear or non-linear
- differentiable or non-smooth
- continuous or integer or mixed-integer
- finite or infinite dimensional
- ...

The great watershed

"The great watershed in optimization isn't between linearity and nonlinearity, but convexity and nonconvexity"

R. Tyrrell Rockafellar

- For convex optimization problems we can efficiently find global minima.
- For non-convex, but smooth problems we can efficiently find local minima.

Literature

- J. Nocedal, S. Wright: Numerical Optimization, Springer, 1999/2006
- P. E. Gill, W. Murray, M. H. Wright: Practical Optimization, Academic Press, 1981
- R. Fletcher, Practical Methods of Optimization, Wiley, 1987
- D. E. Luenberger: Linear and Nonlinear Programming, Addison Wesley, 1984
- S. Boyd, L. Vandenberghe: Convex Optimization, Cambridge University Press, 2004 (PDF freely available at:

<http://web.stanford.edu/~boyd/cvxbook/>

Newton Type Optimization (Unconstrained)

Moritz Diehl

University of Freiburg

(some slide material was provided by W. Bangerth and K. Mombaur)

Aim of Newton type optimization algorithms

$$\min f(x) \quad (x \in R^n)$$

- Find a local minimizer x^* of $f(x)$, i.e. a point satisfying

$$\nabla f(x^*)=0$$

Derivative based algorithms

- **Fundamental underlying structure of most algorithms:**

- choose start value x_0

- for $i=1, \dots$:

- determine direction of search (descent) p

- determine step length α

- new iterate $x_{i+1} = x_i + \alpha p$

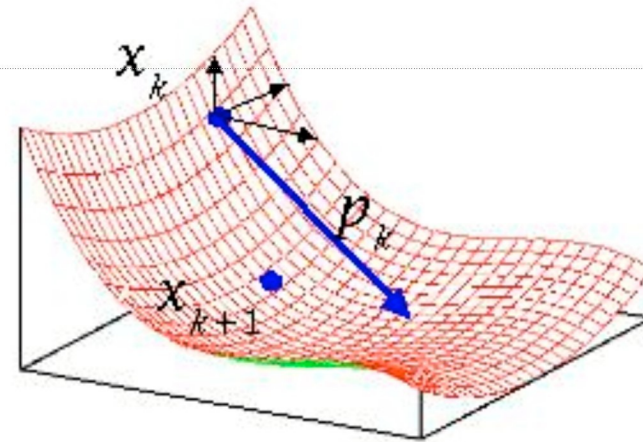
- check convergence

- Optimization algorithms differ in the choice of p and α

Basic algorithm:

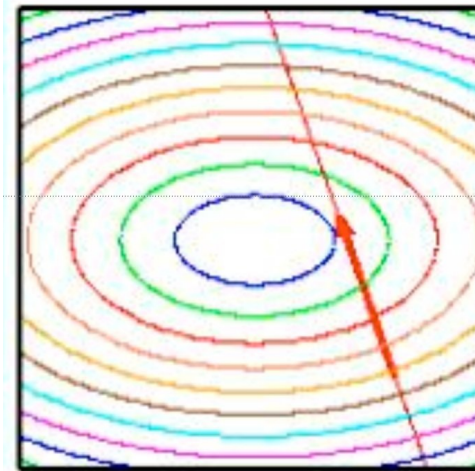
Search direction:

choose descent direction
(f should be decreased)



Step length:

solve 1-d minimization approximately,
satisfy Armijo condition



Computation of step length

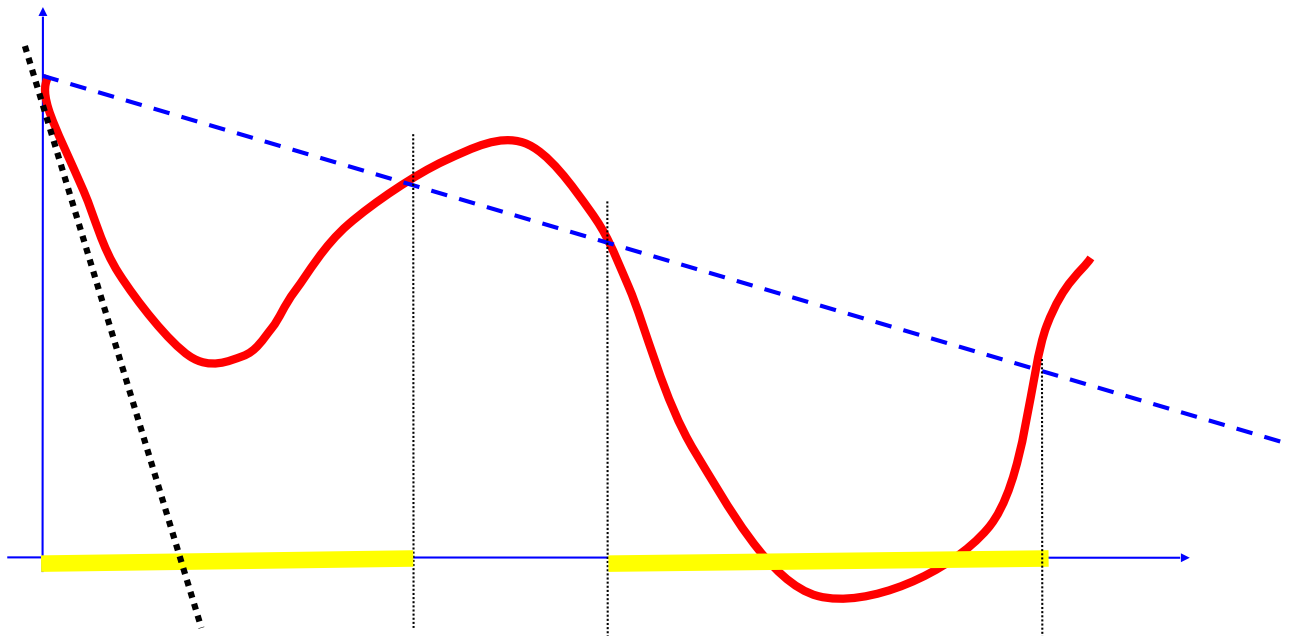
- Dream:

- exact line search: $\alpha^k = \arg \min_{\alpha} f(x^k + \alpha p^k)$

- In practice:

- **inexact line search:** $\alpha^k \approx \arg \min_{\alpha} f(x^k + \alpha p^k)$
- ensure sufficient decrease, e.g. Armijo^αcondition

$$f(x_k + \alpha_k p_k) \leq f(x_k) + c_1 \alpha_k \nabla f_k^T p_k$$



How to compute search direction?

- We discuss three algorithms:
 - Steepest descent method
 - Newton's method
 - Newton type methods

Algorithm 1: Steepest descent method

- Based on first order Taylor series approximation of objective function

$$f(x_k + p_k) = f(x_k) + \underbrace{\nabla f(x_k)^T p_k}_{\text{first order approximation}} + \dots$$

- maximum descent, if

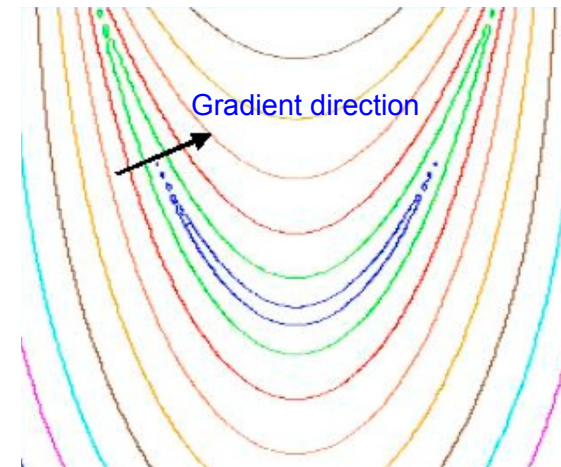
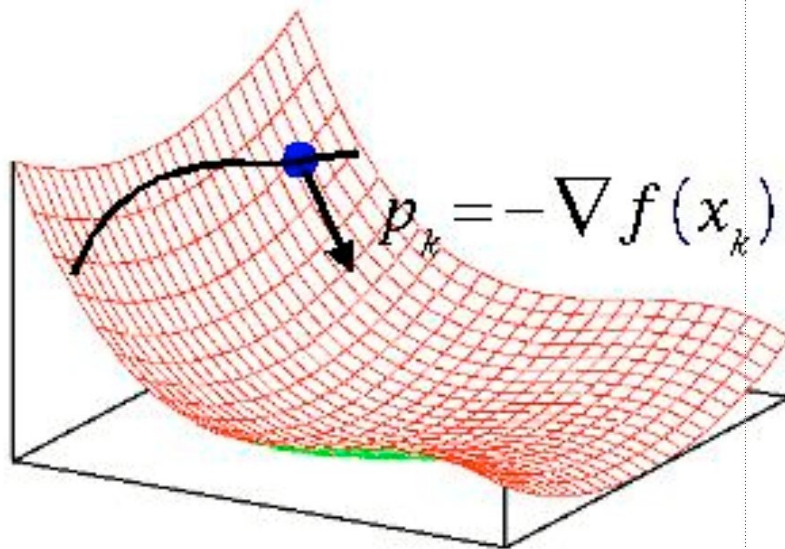
$$\begin{aligned} \frac{\nabla f(x_k)^T p_k}{\|p_k\|} &\rightarrow \min! \\ \Rightarrow p_k &= -\nabla f(x_k) \end{aligned}$$

Steepest descent method

Choose steepest descent search direction, perform (exact) line search:

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

search direction is perpendicular to level sets of $f(x)$



Convergence of steepest descent method

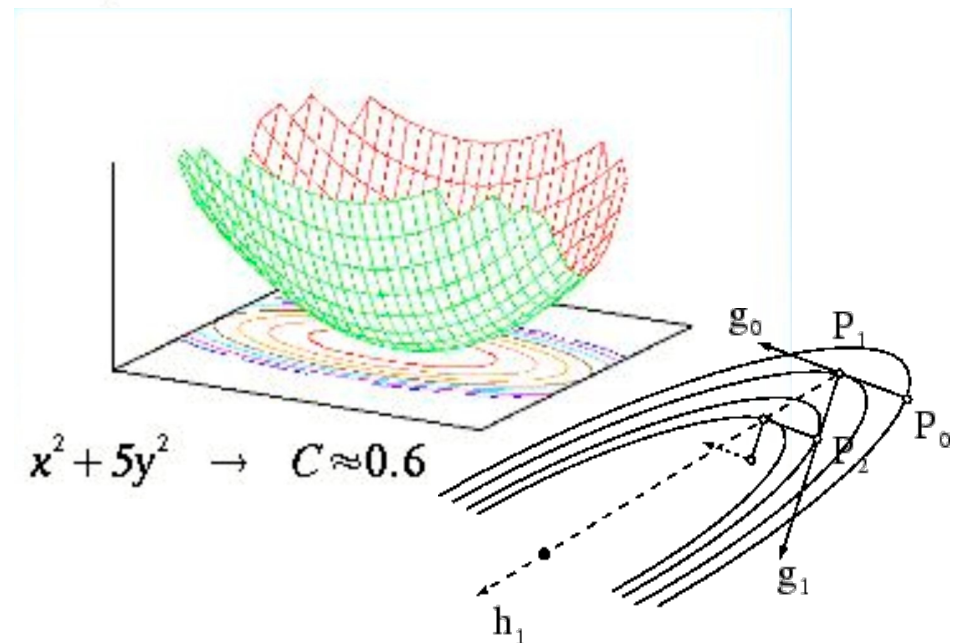
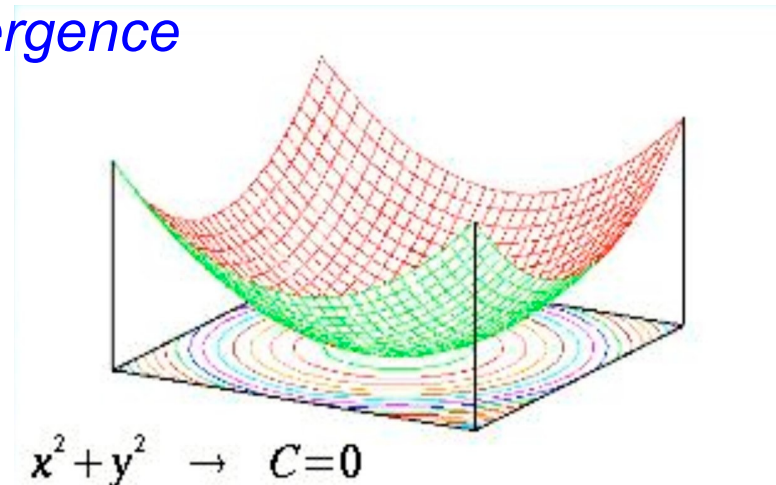
steepest descent method has *linear convergence*

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

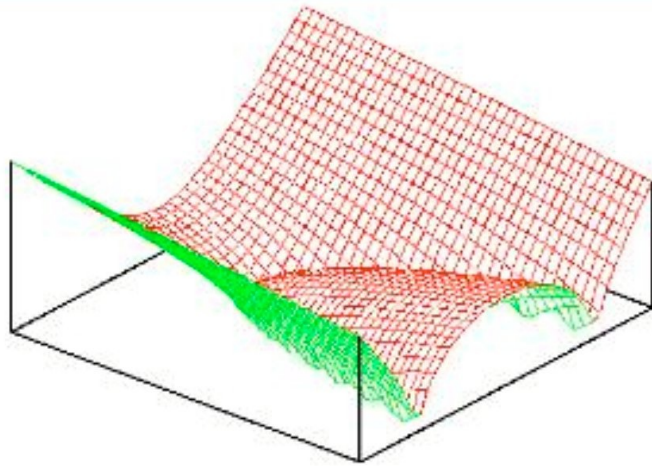
- gain is a fixed factor $C < 1$
- convergence can be very slow if C close to 1

If $f(x) = x^T A x$, A positive definite, λ eigenvalues of A , one can show that

$$\Rightarrow C \approx \frac{\lambda_{\max} - \lambda_{\min}}{\lambda_{\max} + \lambda_{\min}}$$

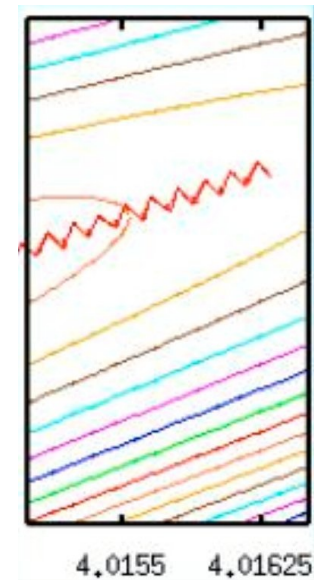
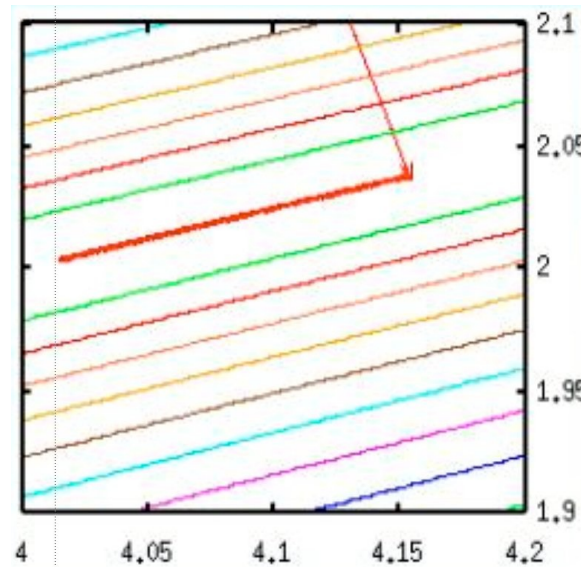
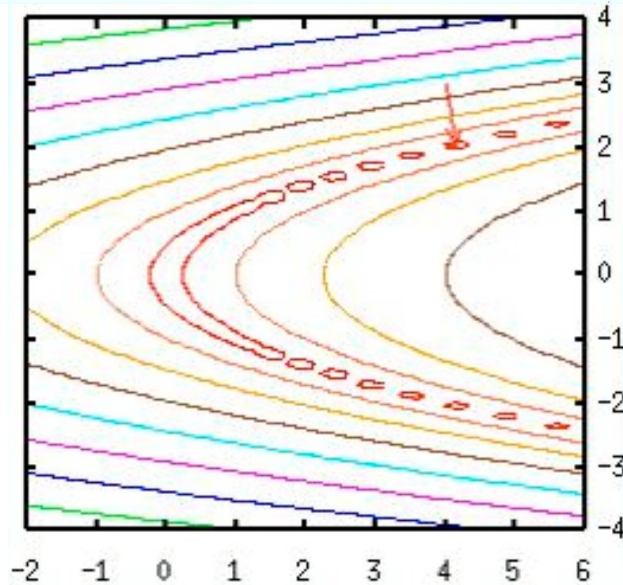


Example - steepest descent method



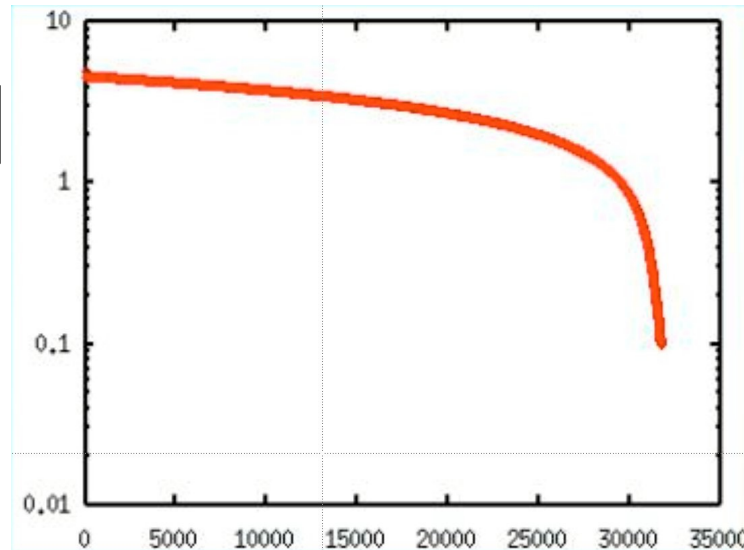
$$f(x, y) = \sqrt[4]{(x - y^2)^2 + \frac{1}{100}} + \frac{1}{100} y^2$$

banana valley function,
global minimum at $x=y=0$



Example - steepest descent method

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



Convergence of steepest descent method:

- needs almost 35.000 iterations to come closer than 0.1 to the solution
- mean value of convergence constant C : 0.99995
- at $(x=4, y=2)$, there holds

$$\lambda_1 = 0.1, \lambda_2 = 268 \quad \Rightarrow \quad C \approx \frac{268 - 0.1}{268 + 0.1} \approx 0.9993$$

Algorithm 2: Newton's Method

- Based on **second order** Taylor series approximation of $f(x)$

$$f(x_k + p_k) = f(x_k) + \underbrace{\nabla f(x_k)^T p_k + \frac{1}{2} p_k^T \nabla^2 f(x_k) p_k}_{\nabla f(x_k)^T p_k + \frac{1}{2} p_k^T \nabla^2 f(x_k) p_k \rightarrow \min!} + \dots$$

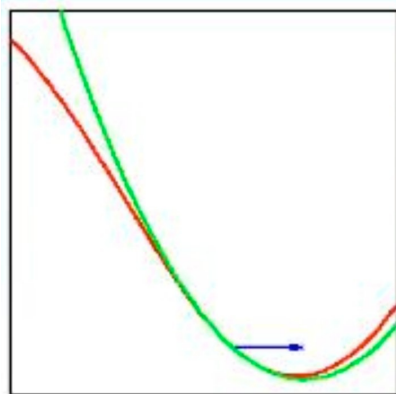
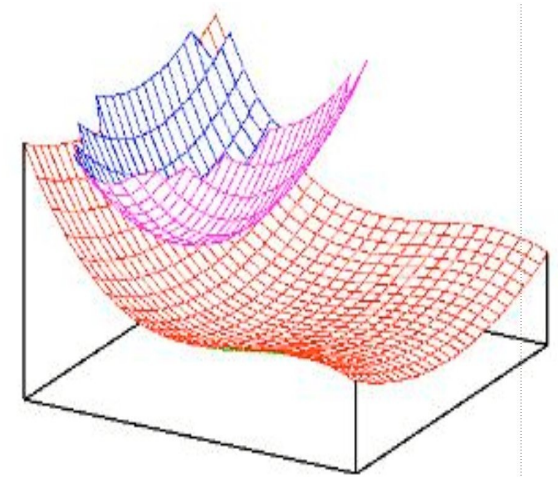
$$\Leftrightarrow \nabla^2 f(x_k) p_k = -\nabla f(x_k)$$

„Newton-Direction“ $p_k = -(\nabla^2 f(x_k))^{-1} \nabla f(x_k)$

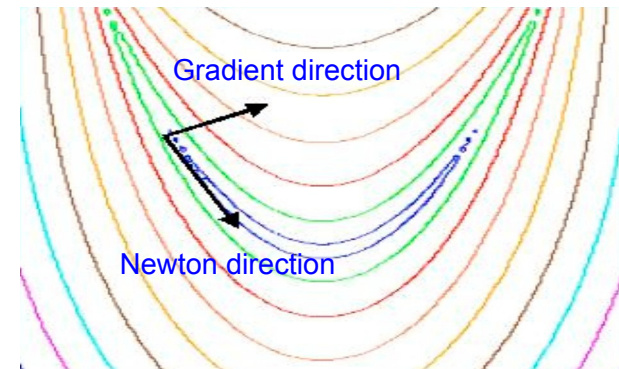
Visualization of Newton's method

p_k minimizes **quadratic approximation** of the objective

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



if quadratic model is good, then take full step with $\alpha^k = 1$



Convergence of Newton's method

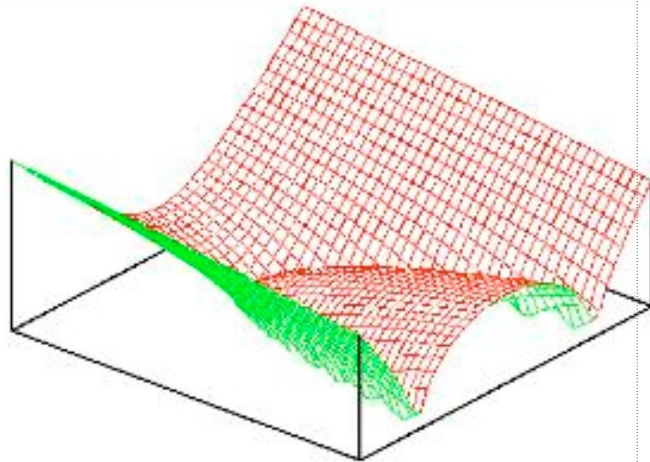
Newton's method has *quadratic convergence*

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

This is ***very fast*** close to a solution:

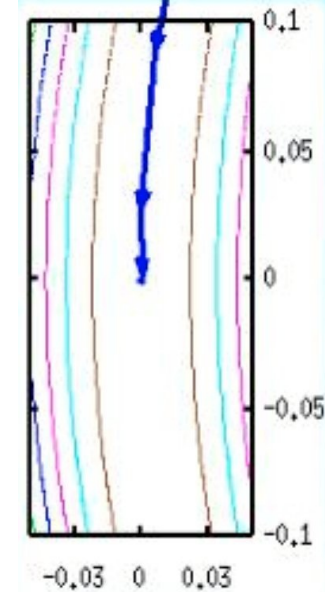
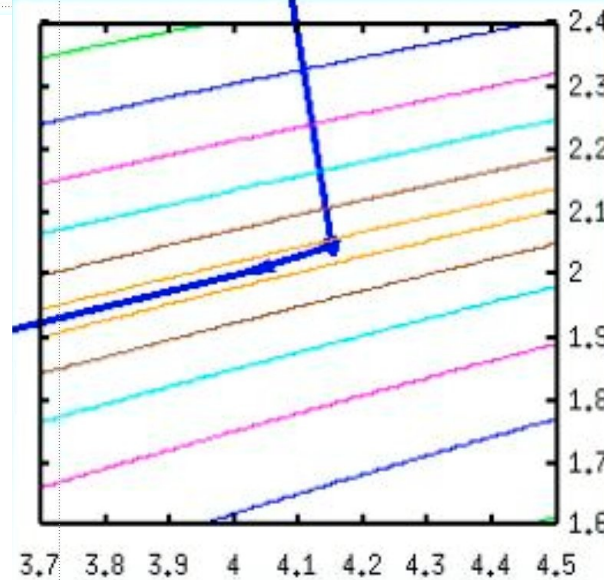
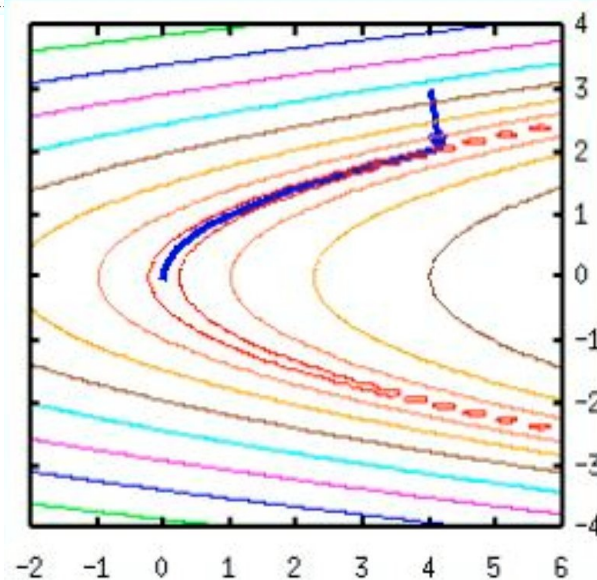
Correct digits double in each iteration!

Example - Newton's method



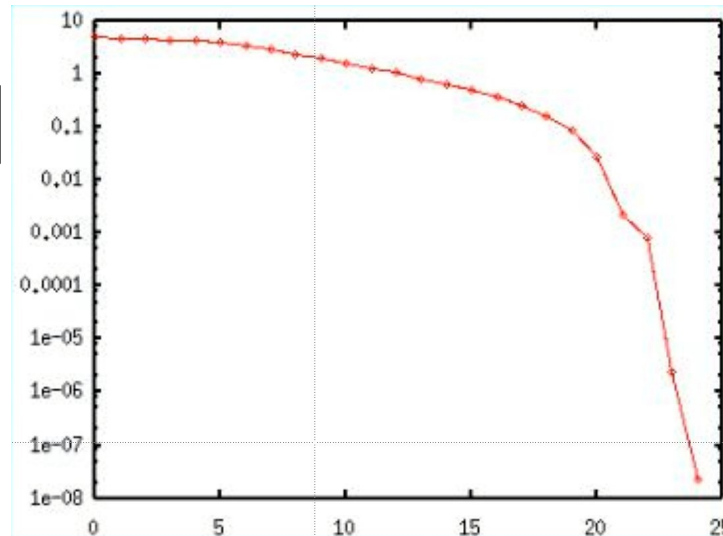
$$f(x, y) = \sqrt[4]{(x - y^2)^2 + \frac{1}{100}} + \frac{1}{100} y^2$$

banana valley function,
global minimum at $x=y=0$



Example - Newton's method

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

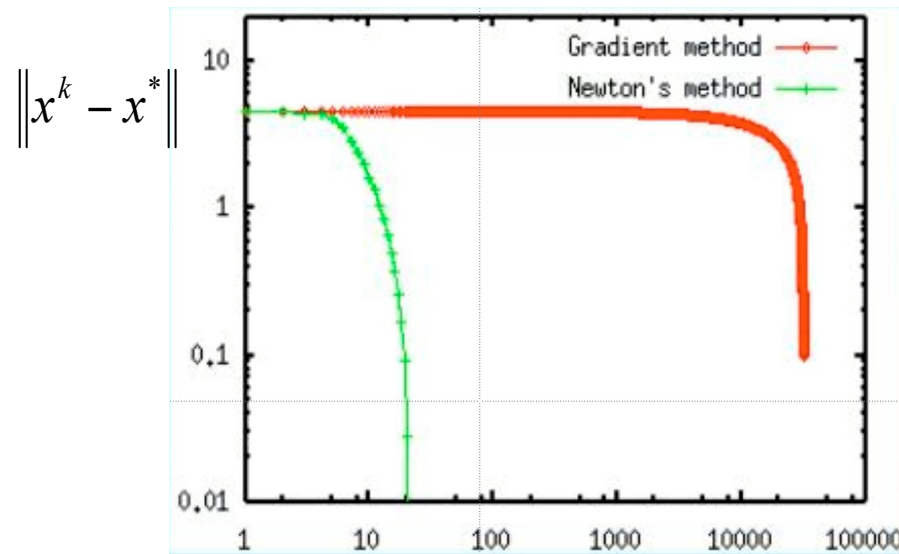


Convergence of Newton's method:

- less than 25 iterations for an accuracy of better than 10^{-7} !
- convergence roughly *linear* for first 15-20 iterations since step length $\alpha_k \neq 1$
- convergence roughly *quadratic* for last iterations with step length

$$\alpha_k = 1$$

Comparison of steepest descent and Newton



For banana valley example:

- Newton's method **much faster** than steepest descent method (factor 1000)
- Newton's method superior due to higher order of convergence
- steepest descent method converges too slowly for practical applications

Algorithm 3: Newton type methods

In practice, evaluation of second derivatives
for the hessian can be difficult!

- approximate hessian matrix $\nabla^2 f(x^k)$
- often methods ensure that the approximation B_k is positive definite

$$x^{k+1} = x^k - B_k^{-1} \nabla f(x^k)$$
$$B_k \approx \nabla^2 f(x^k)$$

methods are collectively known as *Newton type methods*

Newton type variants

Notation:

$$p_k := x_{k+1} - x_k = -B_k^{-1} \nabla f(x^k)$$

- **Steepest Descent:**

$$B_k = I$$

Convergence rate: linear

- **Newton Method:**

$$B_k = \nabla^2 f(x^k)$$

Convergence rate: quadratic

Newton type variants (continued)

- **BFGS** update (Broyden, Fletcher, Goldfarb, Shanno)

$$B_{k+1} = \underset{B_{k+1}}{\operatorname{argmin}} \|B_{k+1}^{-1} - B_k^{-1}\|_{W,F}^2$$
$$\text{s.t. } B_{k+1} p_k = \nabla f(x^{k+1}) - \nabla f(x^k)$$
$$B_{k+1} = B_{k+1}^T$$

Convergence rate: super-linear

- For Least-Squares Problems: **Gauss-Newton Method**

$$f(x) = \frac{1}{2} \|F(x)\|^2 \quad J(x) = \frac{\partial F(x)^T}{\partial x}$$
$$B_k = J(x^k)^T J(x^k)$$

Convergence rate: linear

Summary: Unconstrained Newton Type Optimization

- Aim: find **local minima** of smooth nonlinear problems: $\nabla f(x^*)=0$
- Derivative based methods iterate $x_{i+1} = x_i + \alpha_i p_i$ with
 - **search direction** p_i and **step length** α_i .
 - start at **initial guess** x_0 ,
- Four Newton type methods:
 - steepest descent: intuitive, but slow linear convergence
 - exact Newton's method: very fast quadratic convergence
 - BFGS: fast superlinear convergence
 - Gauss-Newton (only for least-squares): fast linear convergence

Literature

- **J. Nocedal, S. Wright: Numerical Optimization, Springer, 1999/2006**
- P. E. Gill, W. Murray, M. H. Wright: Practical Optimization, Academic Press, 1981
- R. Fletcher, Practical Methods of Optimization, Wiley, 1987
- D. E. Luenberger: Linear and Nonlinear Programming, Addison Wesley, 1984

Constrained Optimization

Moritz Diehl

(some slide material was provided by W. Bangerth, K. Mombaur)

Nonlinear Programming (Problem Class 3)

- General problem formulation:

$$\begin{array}{ll} \min f(x) & f: D \subset \mathbb{R}^n \rightarrow \mathbb{R} \\ \text{s.t. } g(x) = 0 & g: D \subset \mathbb{R}^n \rightarrow \mathbb{R}^l \\ h(x) \geq 0 & h: D \subset \mathbb{R}^n \rightarrow \mathbb{R}^k \end{array}$$

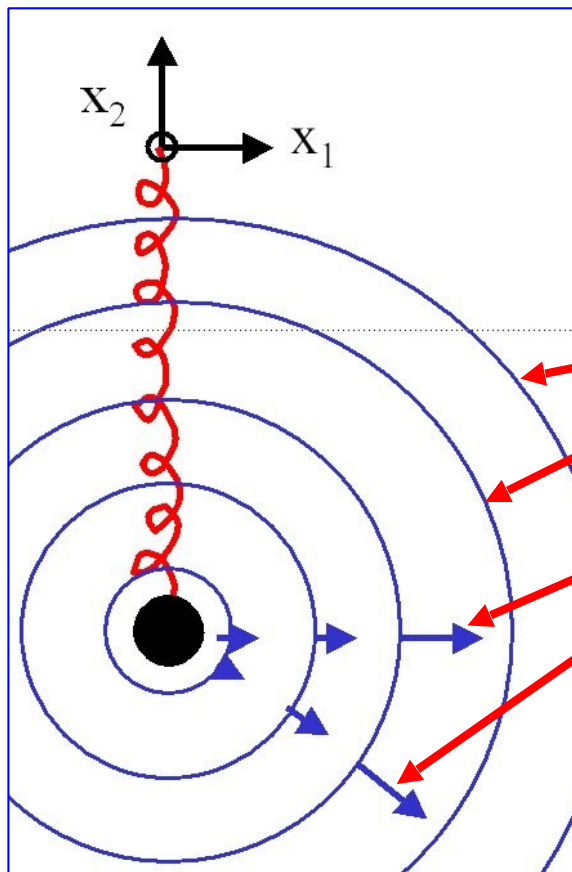
f objective function / cost function

g equality constraints

h inequality constraints

f, g, h shall be smooth (twice differentiable) functions

Recall: ball on a spring without constraints



$$\min_{x \in \mathbb{R}^2} x_1^2 + x_2^2 + mx_2$$

contour lines of $f(x)$

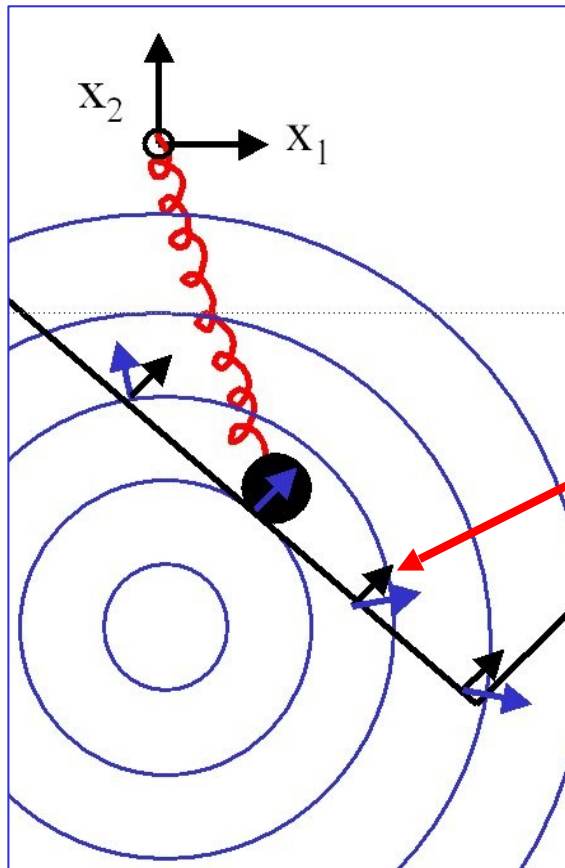
gradient vector

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

unconstrained minimum:

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

Now: ball on a spring with constraints



gradient ∇h_1 of active constraint

inactive constraint h_2

$$\min f(x)$$

$$h_1(x) := 1 + x_1 + x_2 \geq 0$$

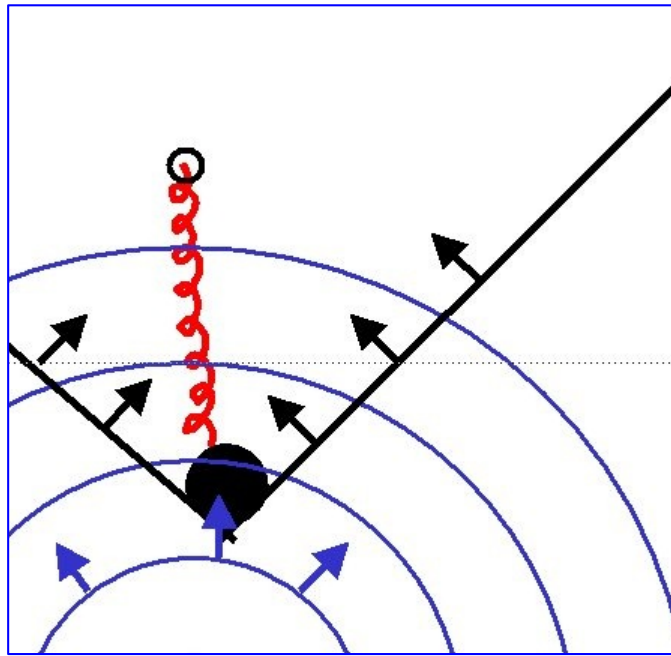
$$h_2(x) := 3 - x_1 + x_2 \geq 0$$

constrained minimum:

$$\nabla f(x^*) = \mu_1 \nabla h_1(x^*)$$

Lagrange multiplier

Ball on a spring with two active constraints



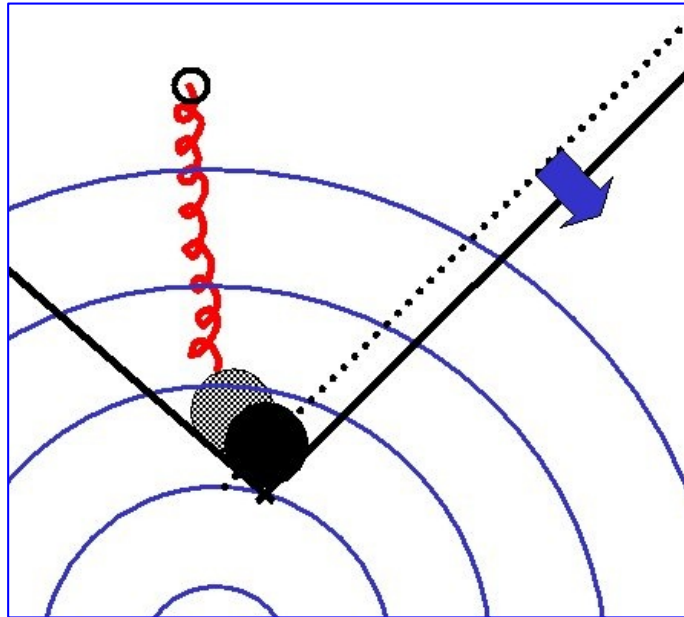
$$\begin{aligned} \min f(x) \\ h_1(x) := 1 + x_1 + x_2 &\geq 0 \\ h_2(x) := 3 - x_1 + x_2 &\geq 0 \end{aligned}$$

„equilibrium of forces“

$$\nabla f(x^*) = \mu_1 \nabla h_1(x^*) + \mu_2 \nabla h_2(x^*) \quad \mu_1, \mu_2 \geq 0$$

„constraint forces“

Multipliers as „shadow prices“



old constraint: $h(x) \geq 0$

new constraint: $h(x) + \varepsilon \geq 0$

What happens if we relax a constraint?
Feasible set becomes bigger,
so new minimum $f(x_\varepsilon^*)$ becomes smaller.
How much would we gain?

$$f(x_\varepsilon^*) \approx f(x^*) - \mu\varepsilon$$

Multipliers show the hidden cost of constraints.

The Lagrangian Function

For constrained problems, introduce modification of objective function:

$$L(x, \lambda, \mu) := f(x^*) - \sum \lambda_i g_i(x) - \sum \mu_i h_i(x)$$

- equality multipliers λ_j may have both signs in a solution
- inequality multipliers μ_j cannot be negative (cf. shadow prices)
- for inactive constraints, multipliers μ_j are zero

Optimality conditions (constrained)

Karush-Kuhn-Tucker necessary conditions (KKT-conditions):

- x^* feasible
- there exist λ^* , μ^* such that

$$\nabla_x L(x^*, \lambda^*, \mu^*) = 0$$

$$(\Leftrightarrow \text{"Equilibrium"} \nabla f = \sum \lambda_i \nabla g_i + \sum \mu_i \nabla h_i)$$

- $\mu^* \geq 0$ holds
- and it holds the complementarity condition

$$\mu^{*T} h(x^*) = 0$$

i.e. $\mu_j^* = 0$ or $h_j(x^*) = 0$ for each i

Sequential Quadratic Programming (SQP)

Constrained problem:

$$\begin{aligned} \min f(x) \\ g(x) &= 0 \\ h(x) &\geq 0 \end{aligned}$$

SQP Idea: Consider successively quadratic approximations of the problem:

$$\begin{aligned} \min_{\Delta x} (\nabla f^k)^T \Delta x + \frac{1}{2} \Delta x^T H^k \Delta x \\ g(x^k) + \nabla g(x^k)^T \Delta x = 0 \\ h(x^k) + \nabla h(x^k)^T \Delta x \geq 0 \end{aligned}$$

SQP method

- if we use the exact hessian of the Lagrangian

$$H = \nabla^2 L(x, \lambda, \mu)$$

this leads to a newton-method for the optimality conditions and feasibility.

- with update-formulas for H^k , we obtain quasi-Newton SQP-methods.
- if we use appropriate update-formulas, we can have superlinear convergence.
- global convergence can be achieved by using a stepsize strategy.

SQP algorithm

0. Start with $k=0$, start value x^0 and $H^0=I$
1. Compute $f(x^k)$, $g(x^k)$, $h(x^k)$, $\nabla f(x^k)$, $\nabla g(x^k)$, $\nabla h(x^k)$
2. If x^k feasible and

$$\|\nabla L(x, \lambda, \mu)\| < \varepsilon$$

then *stop* \Rightarrow convergence achieved

3. Solve quadratic problem and get Δx^k
4. Perform line search and get stepsize t^k
5. Iterate

$$x^{k+1} = x^k + t^k \Delta x^k$$

6. Update hessian
7. $k=k+1$, goto step 1

Conclusions

- Lagrangian function plays important role in constrained optimization
- Lagrange multipliers of inequalities have positive sign
- KKT conditions are necessary optimality conditions
- Newton-type methods iteratively linearise the nonlinear functions and solve simpler (convex) optimisation problems
 - Sequential Quadratic Programming (SQP) is a Newton-type method that iteratively solves quadratic programs (QP) to find a local solution of the non convex problem
 - Nonlinear Interior Point methods are another Newton-type method e.g. implemented in IPOPT



<http://casadi.org>

- A software framework for nonlinear optimization and optimal control
- “Write an efficient optimal control solver in a few lines”
- Implements automatic differentiation (AD) on sparse matrix-valued computational graphs in C++11
- Front-ends to Python, Matlab and Octave
- Supports C code generation
- Back-ends to SUNDIALS, CPLEX, Gurobi, qpOASES, IPOPT, KNITRO, SNOPT, SuperSCS, OSQP, ...
- Developed by Joel Andersson and Joris Gillis



[Andersson, Gillis, Horn, Rawlings, D., Math. Prog. Comp., 2019]

CasADi Demo Solution by Katrin Baumgärtner

Appendix

Professur für Systemtheorie, Regelungstechnik und Optimierung

Forschung:

- Numerische Methoden für die Optimale Steuerung
- Echtzeitoptimierung auf eingebetteten Systemen (***embedded optimization***)
- Modell-prädiktive Regelung (MPC), Methoden und Theorie
- Anwendungen: Windenergie, Leistungselektronik, Motorregelung, Solarthermie, ...
- Entwicklung Freier Software (ACADO, qpOASES, **CasADi**, BLASFEO, **acados**, ...)

Lehre in fünf BSc/MSc Programmen: Embedded Systems Engineering, Mathematik, Mikrosystemtechnik, Nachhaltige Technische Systeme, Informatik:

- Systemtheorie und Regelungstechnik
- Modelling and System Identification
- Numerical Optimization
- Numerical Optimal Control
- Wind Energy Systems
- Race Car Laboratory
- Flight Control Laboratory

