MATH 651: Optimization I

Part 1

Examples of optimization problems

What is an optimization problem?

Mathematically speaking:

Let X be a Banach space; let

$f: X \to R \cup \{+\infty\}$

$g: X \to R^{ne}$

$h: X \to R^{ni}$

be functions on X, find $x \in X$ so that

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

Questions: Under what conditions on X, f, g, h can we guarantee that (i) there is a solution; (ii) the solution is unique; (iii) the solution is stable.

What is an optimization problem?

In practice:

- $x=\{u,y\}$ is a set of design and auxiliary variables that completely describe a physical, chemical, economical model;
- $f(x)$ is an objective function with which we measure how *good* a design is;
- $g(x)$ describes relationships that have to be met exactly, for example the relationship between y and u;
- $h(x)$ describes conditions that must not be exceeded

Then find me that x for which

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

Questions: How do I find this x?

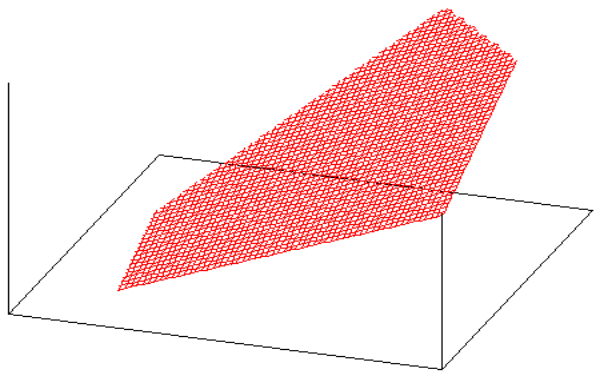
What is an optimization problem?

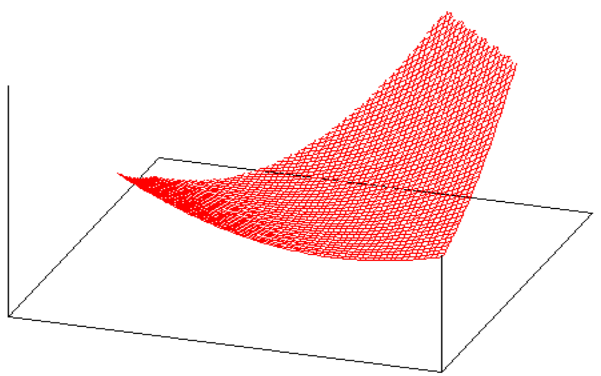
Optimization problems are often subdivided into classes:

Linear	vs.	Nonlinear
Convex	vs.	Nonconvex
Constrained	vs.	Unconstrained
Smooth	vs.	Nonsmooth
With derivatives	vs.	Derivativefree
Continuous	vs.	Discrete
Algebraic	vs.	ODE/PDE

Depending on which class an actual problem falls into, one method or another needs to be chosen.

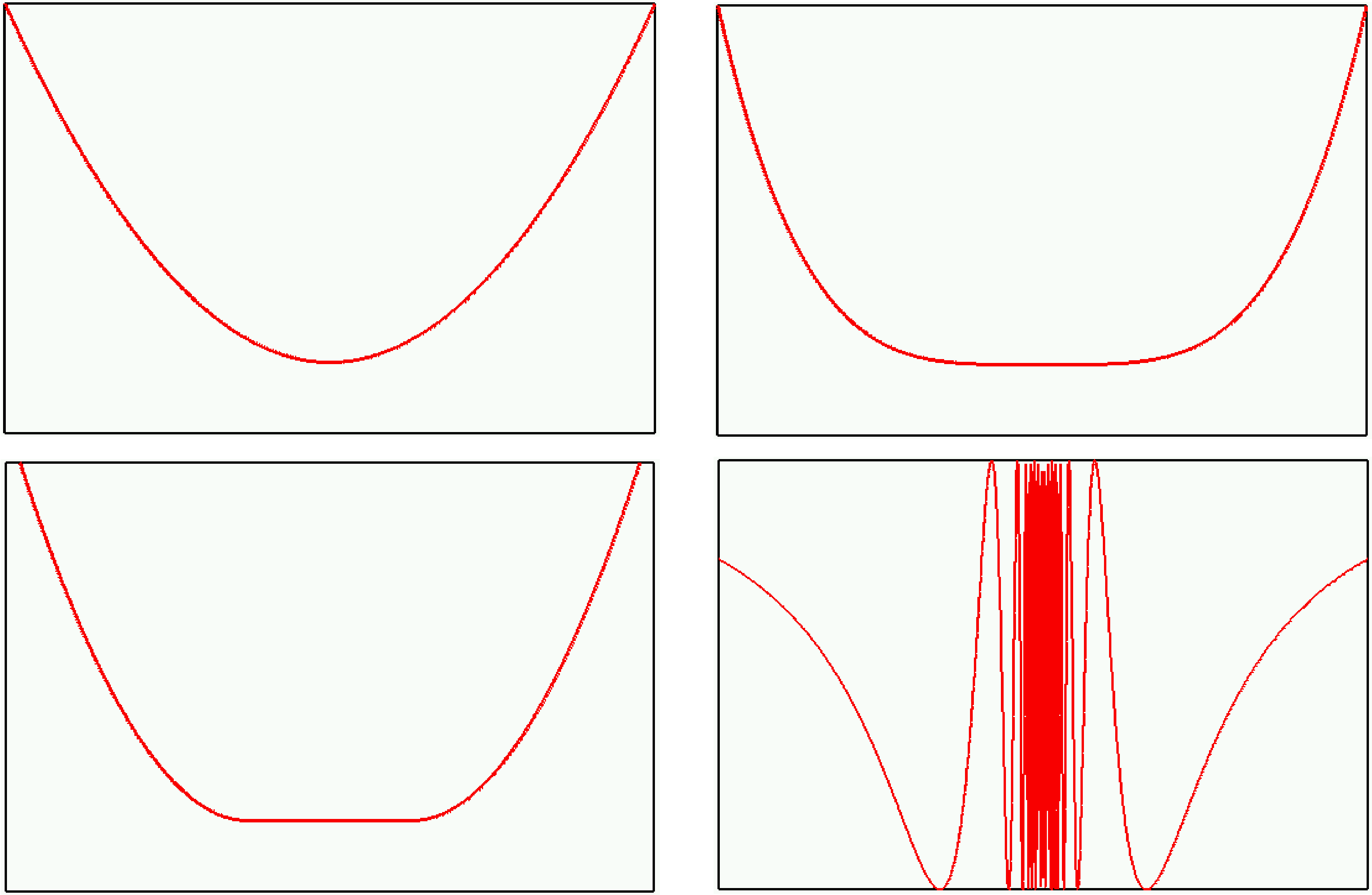
Examples





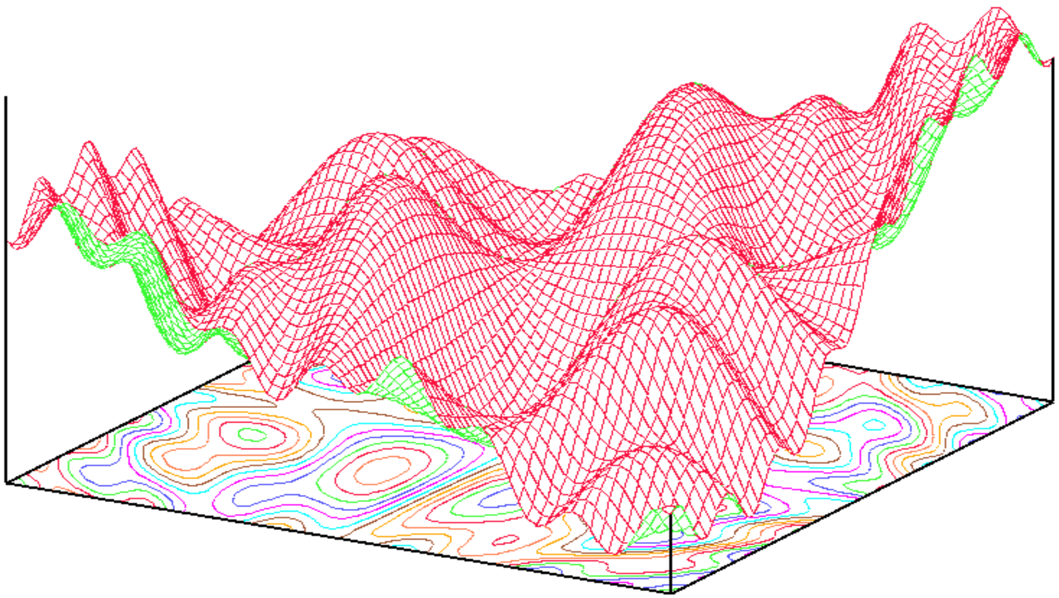
Linear and nonlinear functions *f(x)*
on a domain bounded by linear inequalities

Examples



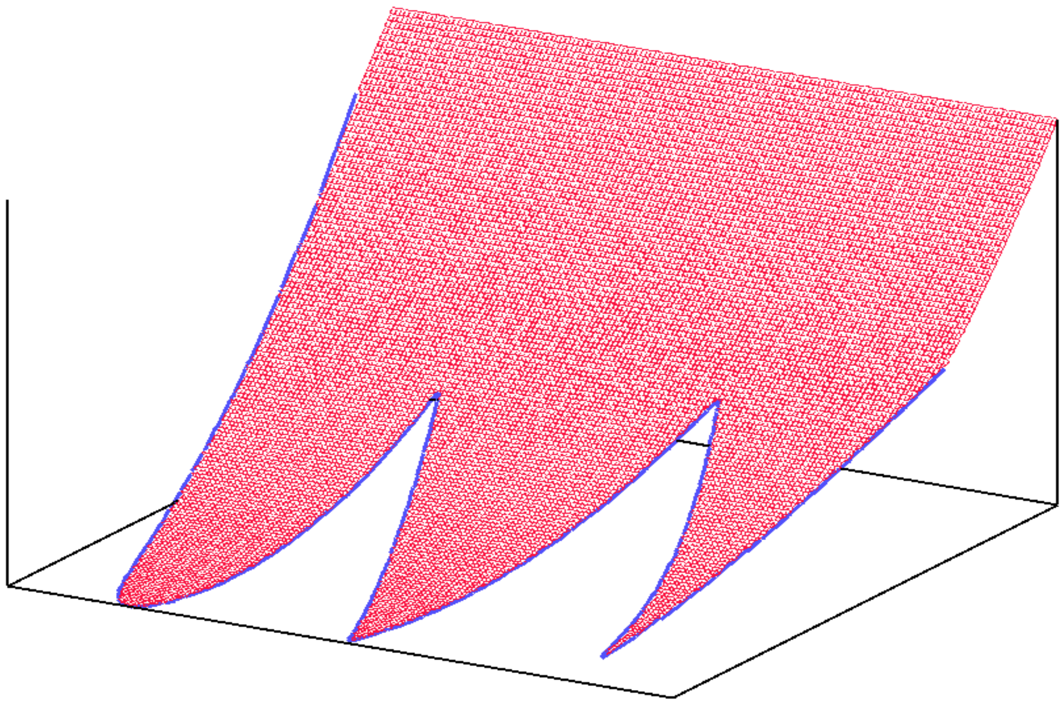
Strictly convex, convex, and nonconvex functions *f(x)*

Examples



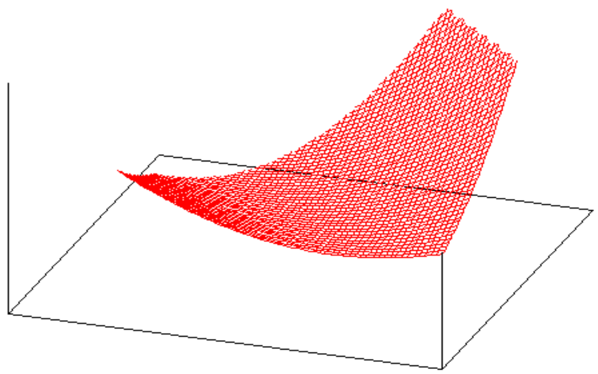
Another non-convex function with many (local) optima.
We may want to find the one *global* optimum.

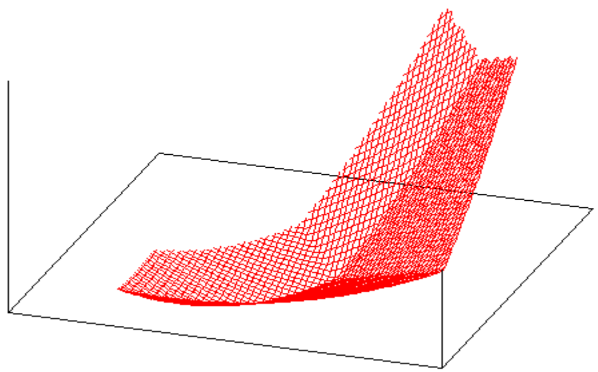
Examples



Optima in the presence of (nonsmooth) constraints.

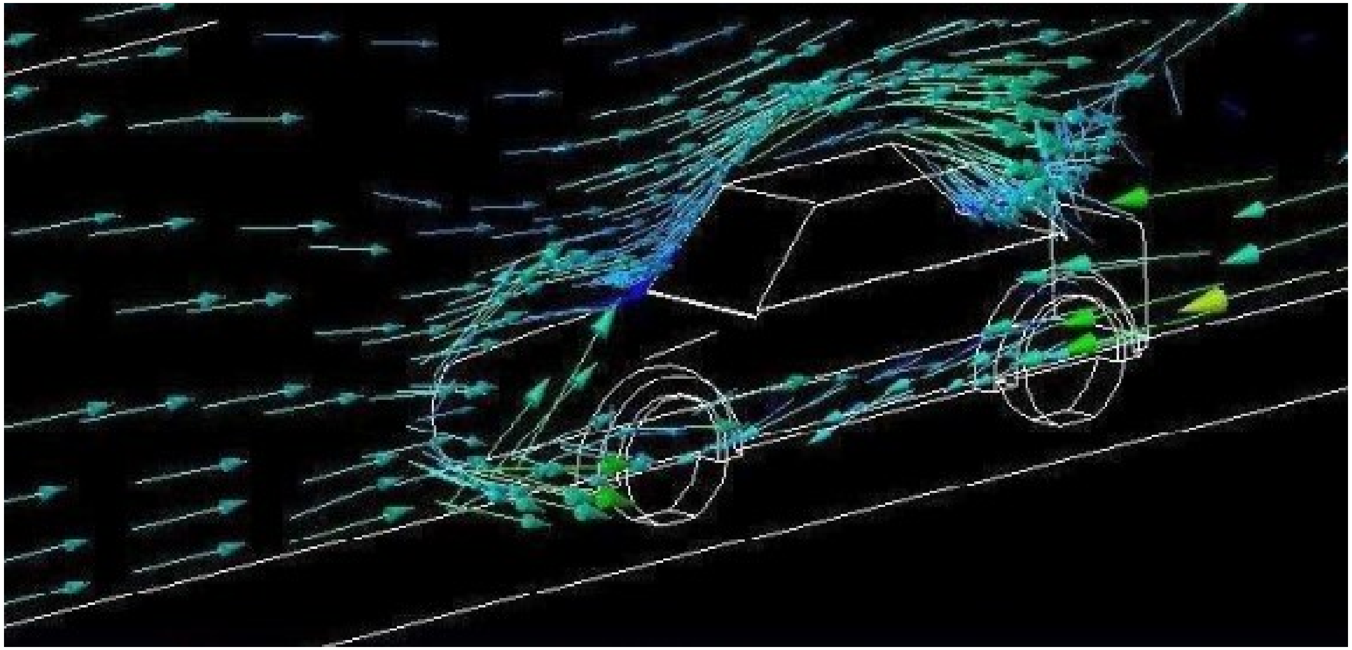
Examples





Smooth and non-smooth nonlinear functions.

Applications: The drag coefficient of a car



Mathematical description:

$x=\{u,y\}$: u are the design parameters (e.g. the *shape* of the car)
y is the flow field around the car

$f(x)$: the drag force that results from the flow field

$g(x)=y-q(u)=0$:
constraints that come from the fact that there is a flow field $y=q(u)$ for each design. y may, for example, satisfy the Navier-Stokes equations

Applications: The drag coefficient of a car

Inequality constraints:

(expected sales price – profit margin) - cost(u) ≥ 0

volume(u) – volume(me, my wife, and her bags) ≥ 0

max(forces exerted by y on the frame)
- material stiffness * safety factor ≥ 0

legal margins(u) ≥ 0

Applications: The drag coefficient of a car

Analysis:

linearity:	*f(x)* may be linear *g(x)* is certainly nonlinear (Navier-Stokes equations) *h(x)* may be nonlinear
convexity:	??
constrained:	yes
smooth:	*f(x)* yes *g(x)* yes *h(x)* some yes, some no
derivatives:	available, but probably hard to compute in practice
continuous:	yes, not discrete
ODE/PDE:	yes, not just algebraic

Applications: The drag coefficient of a car

Remark:

In the formulation as shown, the objective function was of the form

$$f(x) = c_d(y)$$

In practice, one often is willing to trade efficiency for cost, i.e. we are willing to accept a slightly higher drag coefficient if the cost is smaller. This leads to objective functions of the form

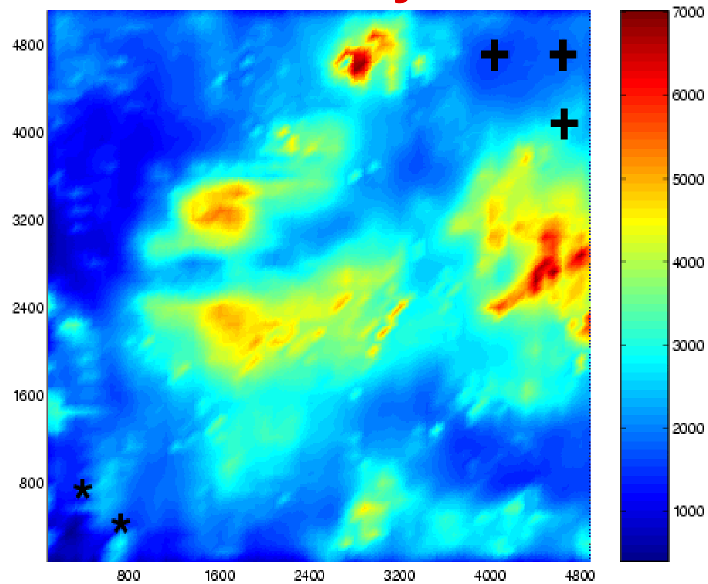
$$f(x) = c_d(y) + a\,\mathrm{cost}(u)$$

or

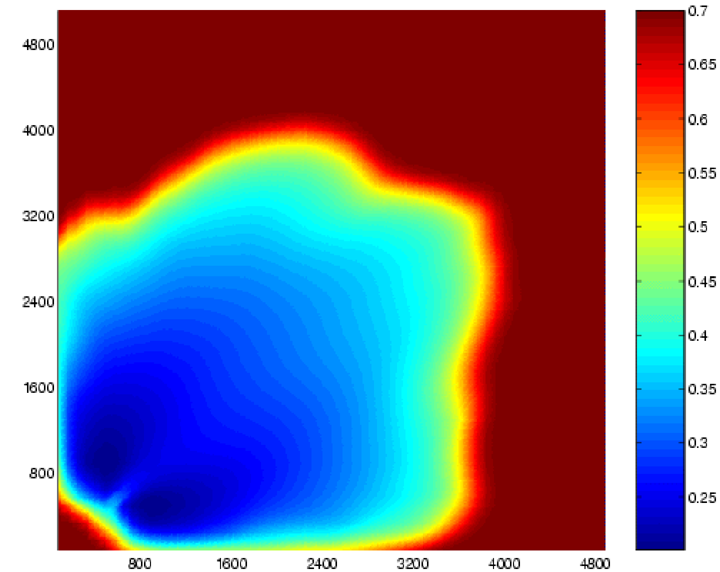
$$f(x) = c_d(y) + a[\mathrm{cost}(u)]^2$$

Applications: Optimal oil production strategies

Permeability field



Oil saturation



Mathematical description:

x={u,y}: *u* are the pumping rates at injection/production wells
y is the flow field (pressures/velocities)

f(x): the cost of production and injection minus sales price of oil integrated over lifetime of reservoir (or -NPV)

g(x)=y-q(u)=0:
constraints that come from the fact that there is a flow field *y=q(u)* for each *u*. *y* may, for example, satisfy the multiphase porous media flow equations

Applications: Optimal oil production strategies

Inequality constraints $h(x) \geq 0$:

$u_{imax} - u_i \geq 0$:

Pumps have a maximal pumping rate/pressure

produced_oil(T)/available_oil(0) – $c \geq 0$:

Legislative requirement to produce at least a certain fraction

c - water_cut(t) ≥ 0 (for all times t):

It is inefficient to produce too much water

pressure – $d \geq 0$ (for all times and locations):

Keeps the reservoir from collapsing

Applications: Optimal oil production strategies

Analysis:

linearity:	$f(x)$ is nonlinear $g(x)$ is certainly nonlinear $h(x)$ may be nonlinear
convexity:	no
constrained:	yes
smooth:	$f(x)$ yes $g(x)$ yes $h(x)$ yes
derivatives:	available, but probably hard to compute in practice
continuous:	yes, not discrete
ODE/PDE:	yes, not just algebraic

Applications: Switching lights at an intersection

Mathematical description:

$x = \{T, t_i^1, t_i^2\}$: round-trip time T for the stop light system, switch-green and switch-red times for all lights i

$f(x)$: number of cars that can pass the intersection per hour;

Note: unknown as a function, but we can measure it

Applications: Switching lights at an intersection

Inequality constraints $h(x) \geq 0$:

$300 - T \geq 0$:

No more than 5 minutes of round-trip time, so that people don't have to wait for too long

$t_{2i} - t_{1i} - 5 \geq 0$ (for all lights i):

At least 5 seconds of green for everyone

$t_{1(i+1)} - t_{2i} - 5 \geq 0$:

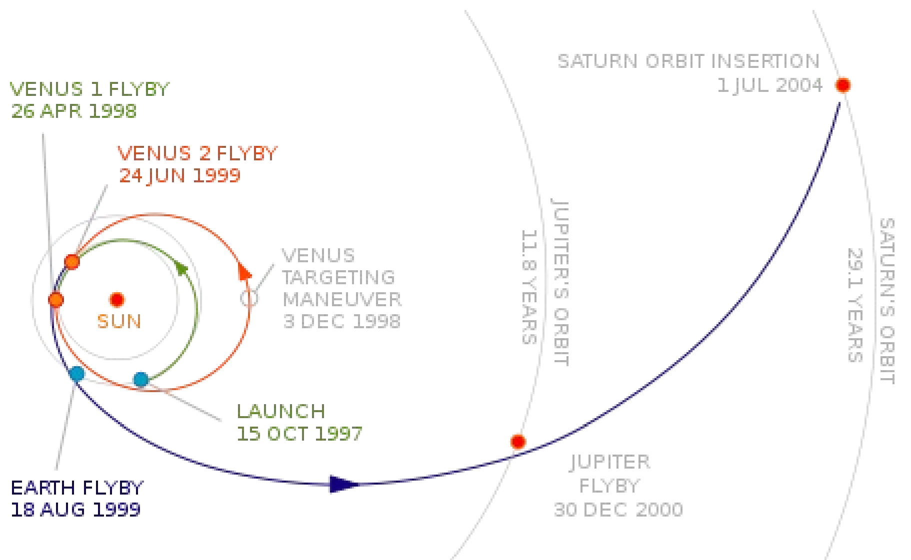
At least 5 seconds of all-red between different greens

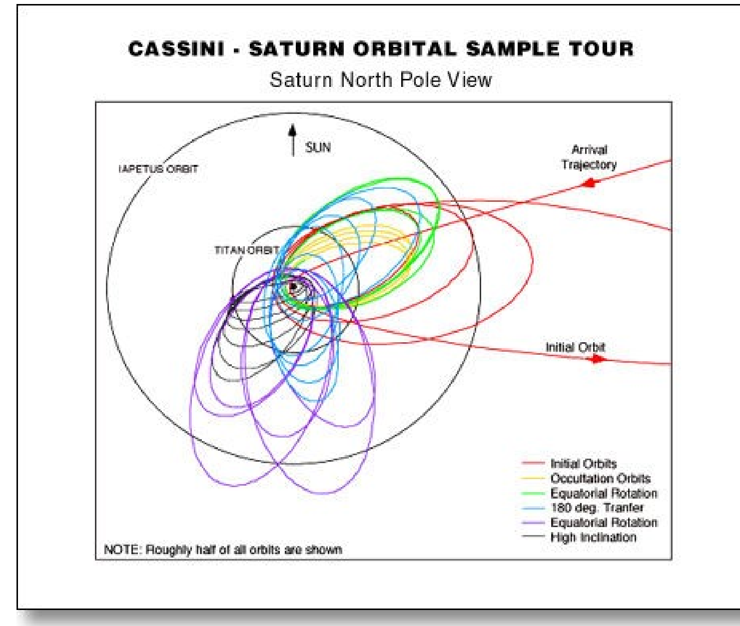
Applications: Switching lights at an intersection

Analysis:

linearity:	$f(x)$??
	$h(x)$ is linear
convexity:	??
constrained:	yes
smooth:	$f(x)$ *??*
	$h(x)$ yes
derivatives:	not available
continuous:	yes, not discrete
ODE/PDE:	no

Applications: Trajectory planning





Mathematical description:

$x=\{y(t),u(t)\}$:	position of spacecraft and thrust vector at time t
$f(x)=\int_0^T \lvert u(t)\rvert dt$	minimize fuel consumption
$m\ddot{y}(t)-u(t)=0$	Newton's law
$\lvert y(t)\rvert-d_0\geq 0$	Do not get too close to the sun
$u_{\max}-\lvert u(t)\rvert\geq 0$	Only limited thrust available

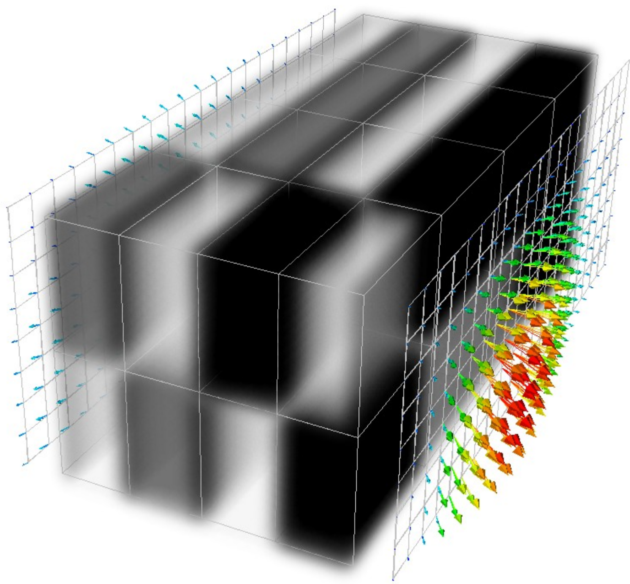
Applications: Trajectory planning

Analysis:

linearity:	*f(x)* is nonlinear *g(x)* is linear *h(x)* is nonlinear
convexity:	no
constrained:	yes
smooth:	yes, here
derivatives:	computable
continuous:	yes, not discrete
ODE/PDE:	yes

Note: Trajectory planning problems are often called *optimal control*.

Applications: Economic games



Setup: A container loaded with materials q_i and a nuclear device produces radiation *r(q)*.

Choosing detector locations s_i will allow us to flag containers that produce radiation *r(q)* with probability *p(r(q),s)*.

Goal: *Choose the best detector locations!*

Mathematical description:

$x=\{q_i, s_i\}$: cargo composition, detector locations

$f(x) = \min_{q_i} p(r(q), s) \rightarrow \max$ probability of detection

$c_0 - \text{cost}(s) \geq 0$ detection strategy must be feasible

$d_0 - \text{cost}(q) \geq 0$ bad guys have limited resources

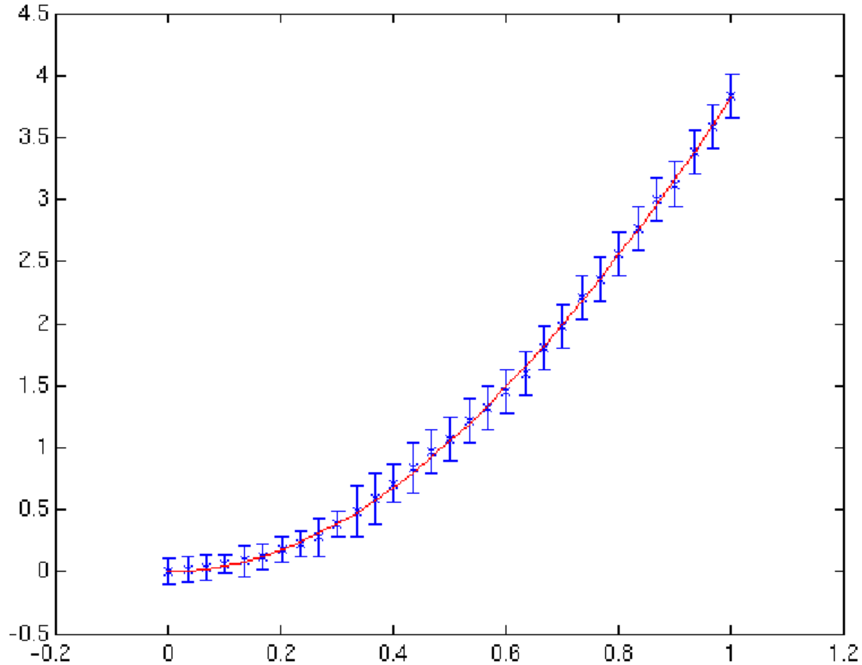
Applications: Economic games

Analysis:

linearity:	$f(x)$ is nonlinear $h(x)$ is nonlinear
convexity:	??
constrained:	yes
smooth:	in general not
derivatives:	sometimes
continuous:	yes, but often have discrete components
ODE/PDE:	no

Note: Games often have elegant solutions in the form of (Nash) equilibria.

Applications: Data fitting 1



Mathematical description:

$x=\{a,b\}$: parameters for the model $y(t) = \frac{1}{a} \log\cosh(\sqrt{ab}\,t)$

$f(x)=1/N \sum_i |y_i - y(t_i)|^2$:

mean square difference between predicted value and actual measurement

Applications: Data fitting 1

Analysis:

linearity: $f(x)$ is nonlinear

convexity: ?? (probably yes)

constrained: no

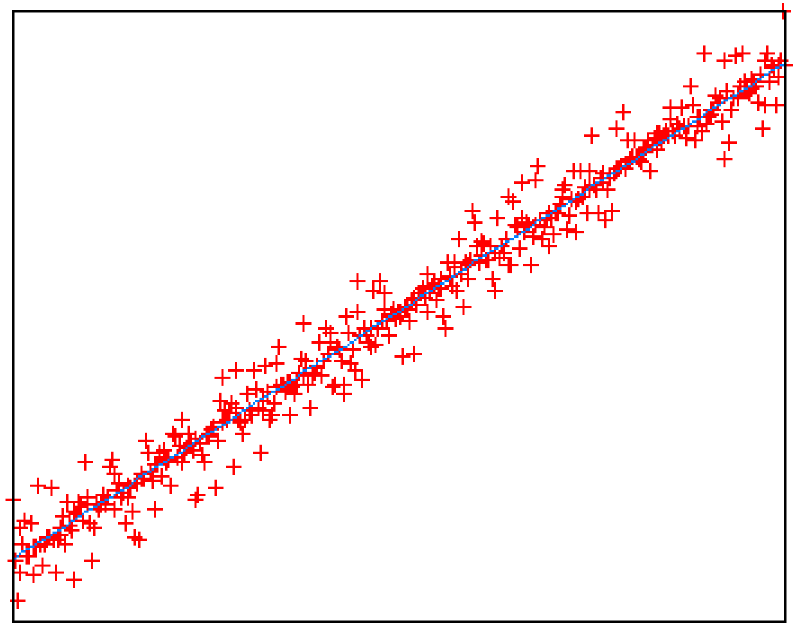
smooth: yes

derivatives: available, and easy to compute in practice

continuous: yes, not discrete

ODE/PDE: no, algebraic

Applications: Data fitting 2



Mathematical description:

$x=\{a,b\}$: parameters for the model $y(t)=at+b$

$f(x)=1/N \sum_i |y_i-y(t_i)|^2$:

mean square difference between predicted value and actual measurement

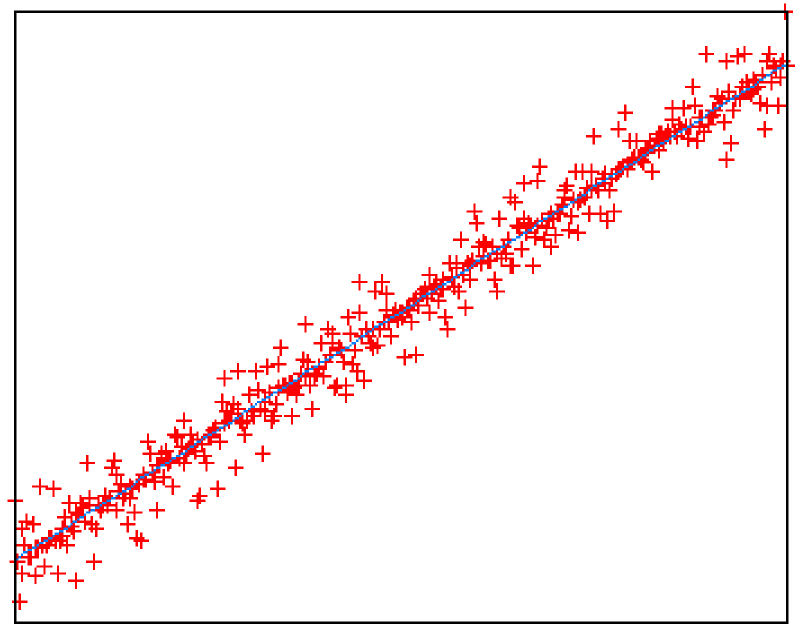
Applications: Data fitting 2

Analysis:

linearity:	*f(x)* is quadratic
Convexity:	yes
constrained:	no
smooth:	yes
derivatives:	available, and easy to compute in practice
continuous:	yes, not discrete
ODE/PDE:	no, algebraic

Note: Quadratic optimization problems (even if we have linear constraints) are easy to solve!

Applications: Data fitting 3



Mathematical description:

$x=\{a,b\}$: parameters for the model $y(t)=at+b$

$f(x)=1/N \sum_i |y_i-y(t_i)|$:

mean *absolute* difference between predicted value and actual measurement

Applications: Data fitting 3

Analysis:

linearity: *f(x)* is nonlinear

Convexity: yes

constrained: no

smooth: no!

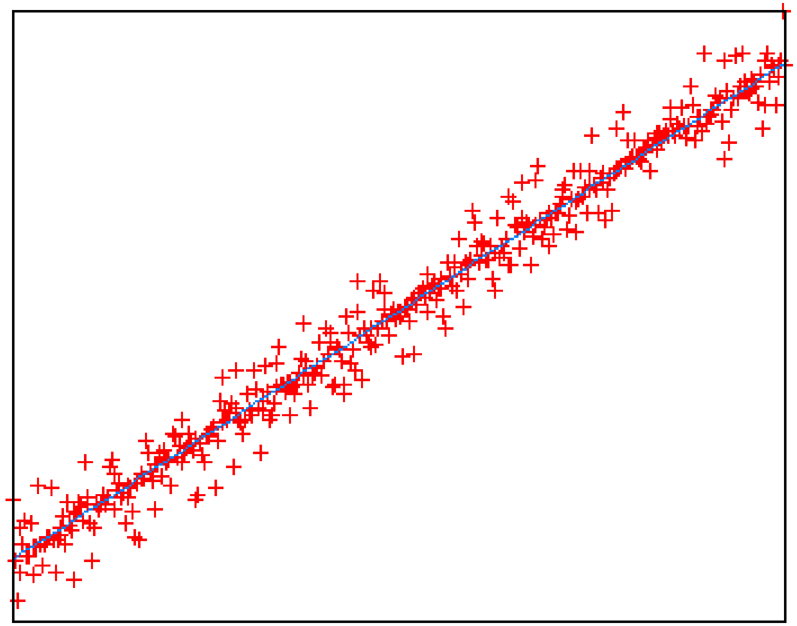
derivatives: not differentiable

continuous: yes, not discrete

ODE/PDE: no, algebraic

Note: Non-smooth problems are really hard to solve!

Applications: Data fitting 3, revisited



Mathematical description:

$x=\{a,b, s_i\}$: parameters for the model $y(t)=at+b$

"slack" variables s_i

$f(x)=1/N \sum_i s_i \rightarrow$ min!

$s_i - |y_i - y(t_i)| \geq 0$

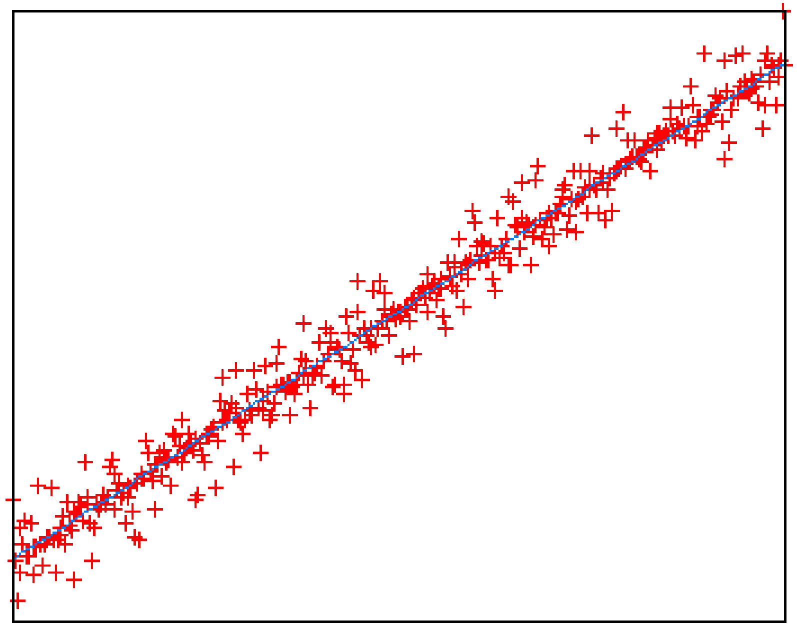
Applications: Data fitting 3, revisited

Analysis:

linearity:	$f(x)$ is linear, $h(x)$ is not linear
Convexity:	yes
constrained:	yes
smooth:	no!
derivatives:	not differentiable
continuous:	yes, not discrete
ODE/PDE:	no, algebraic

Note: Non-smooth problems are really hard to solve!

Applications: Data fitting 3, re-revisited



Mathematical description:

$x=\{a,b, s_i\}$: parameters for the model $y(t) = at+b$

"slack" variables s_i

$f(x)=1/N \sum_i s_i \rightarrow$ min!

$\cancel{s_i - |y_i-y(t_i)| \geq 0}$

$s_i - (y_i-y(t_i)) \geq 0$

$s_i + (y_i-y(t_i)) \geq 0$

Applications: Data fitting 3, re-revisited

Analysis:

linearity:	$f(x)$ is linear, $h(x)$ is now also linear
Convexity:	yes
constrained:	yes
smooth:	yes
derivatives:	yes
continuous:	yes, not discrete
ODE/PDE:	no, algebraic

Note: Linear problems with linear constraints are simple to solve!

Applications: Traveling salesman

Task: Find the shortest tour through N cities with mutual distances d_{ij}.

(Here: the 15 biggest cities of Germany; there are 43,589,145,600 possible tours through all these cities.)

Mathematical description:

$x=\{c_i\}$: the index of the ith city on our trip, $i=1\ldots N$

$f(x)=\sum_i d_{c_i c_{i+1}}$

$c_i \neq c_j$ for $i \neq j$ no city is visited twice (alternatively: $c_i c_j \geq 1$)

Applications: Traveling salesman

Analysis:

linearity:	$f(x)$ is linear, $h(x)$ is nonlinear
Convexity:	meaningless
constrained:	yes
smooth:	meaningless
derivatives:	meaningless
continuous:	discrete: $x \in X \subset \{1,2,\ldots,N\}^N$
ODE/PDE:	no, algebraic

Note: Integer problems (combinatorial problems) are often exceedingly complicated to solve!

Part 2

Minima, minimizers, sufficient and necessary conditions

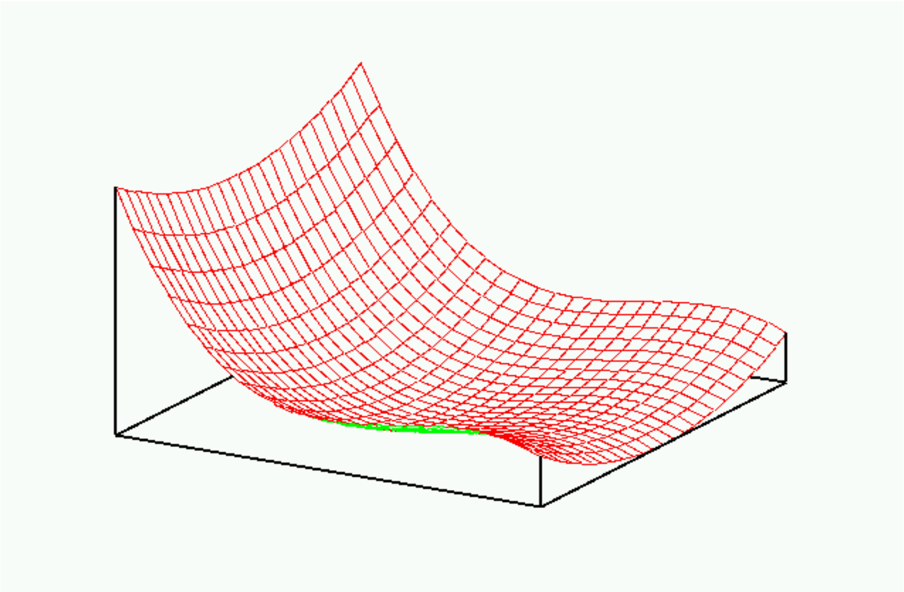
Part 3

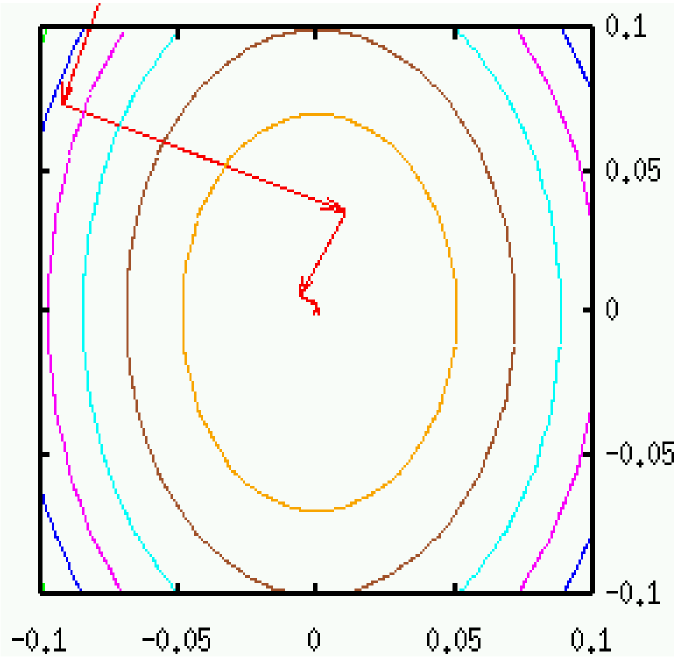
Metrics of algorithmic complexity

Outline of optimization algorithms

All algorithms to find minimia 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 minimia of $f(x)$ do so iteratively:

- start at a point x_0
- for $k=1,2,\ldots,$:
  . 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:

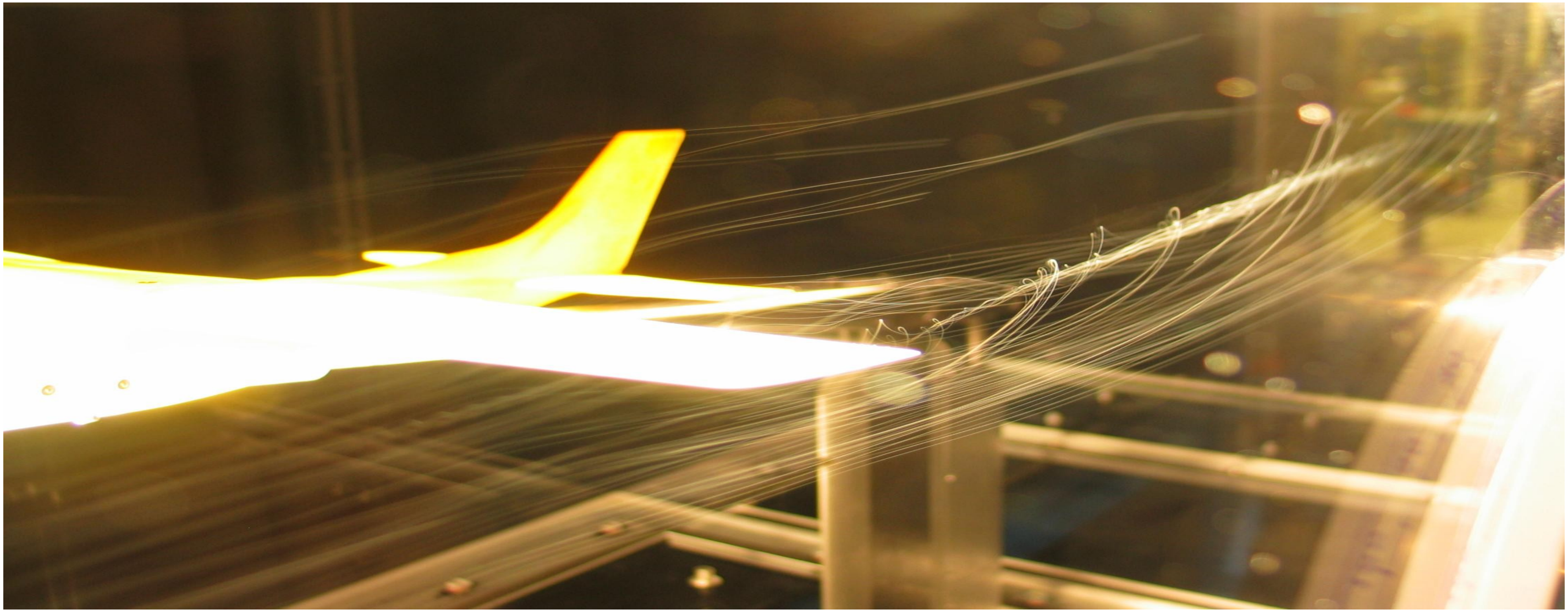
- If x^* is the minimizer that we are seeking, does $x_k \rightarrow x^*$?
- How many iterations does it take for $\|x_k - x^*\| \leq \epsilon$?
- How expensive is every iteration?

How expensive is every iteration?

•Most of the time, the cost of optimization algorithms is dominated by the cost of evaluating $f(x)$, $g(x)$, $h(x)$ and their derivatives:

•**Traffic light example:** Evaluating $f(x)$ might require us to sit at an intersection for an hour, counting cars

•**Designing air foils:** Coming up with an improved wing design and testing it in a wind tunnel costs millions of dollars.



How expensive is every iteration?

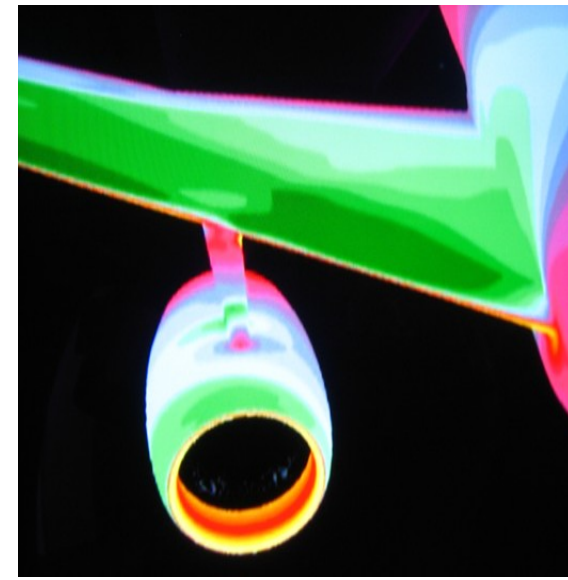
Example: Boeing wing design

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

- The gradient (steepest descent) method requires 1 evaluation of $\nabla f(\cdot)$ per iteration
- The Newton method requires 1 evaluation of $\nabla f(\cdot)$ and one 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 $\nabla f(\cdot)$

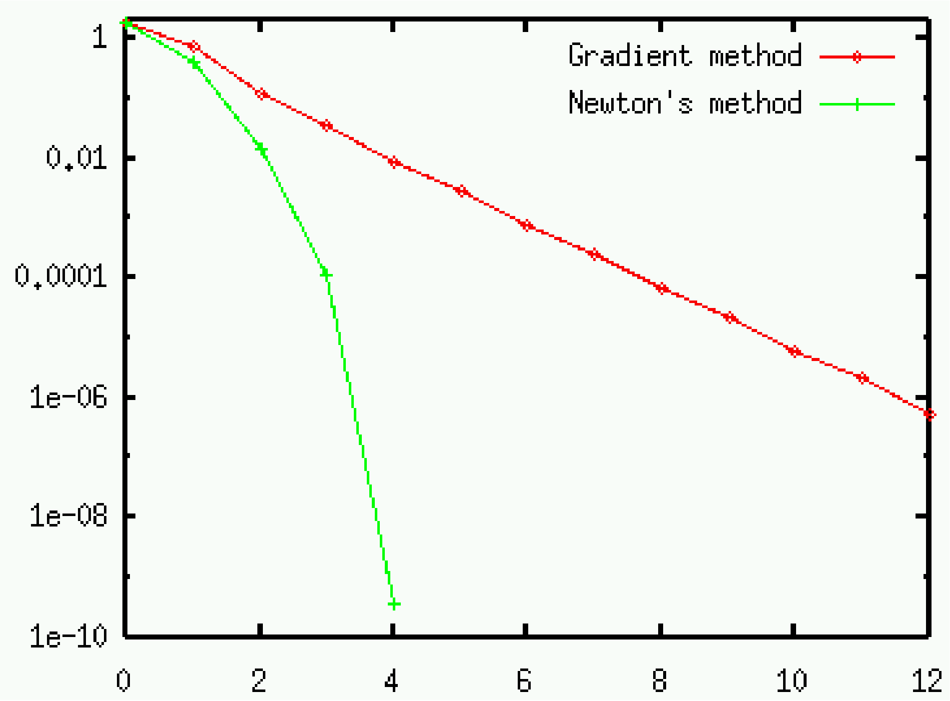
To determine the step length α_k

- 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

How many iterations do we need?

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

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

More generally, a sequence of numbers $a_k \to 0$ is called of order *s* if

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

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

Specifically:

If *s=1*, then the sequence is called *linearly convergent*. In this case, convergence requires *C<1*. In a singly logarithmic plot, linearly convergent sequences are straight lines.

If *s=2*, then the sequence is called *quadratically convergent*.

If *1<s<2*, then the sequence is called *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: Linearly convergent sequences can converge very slowly if C is close to one. Generally, linear convergence is considered *slow* and we will want to avoid algorithms that have linear convergence.

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: 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

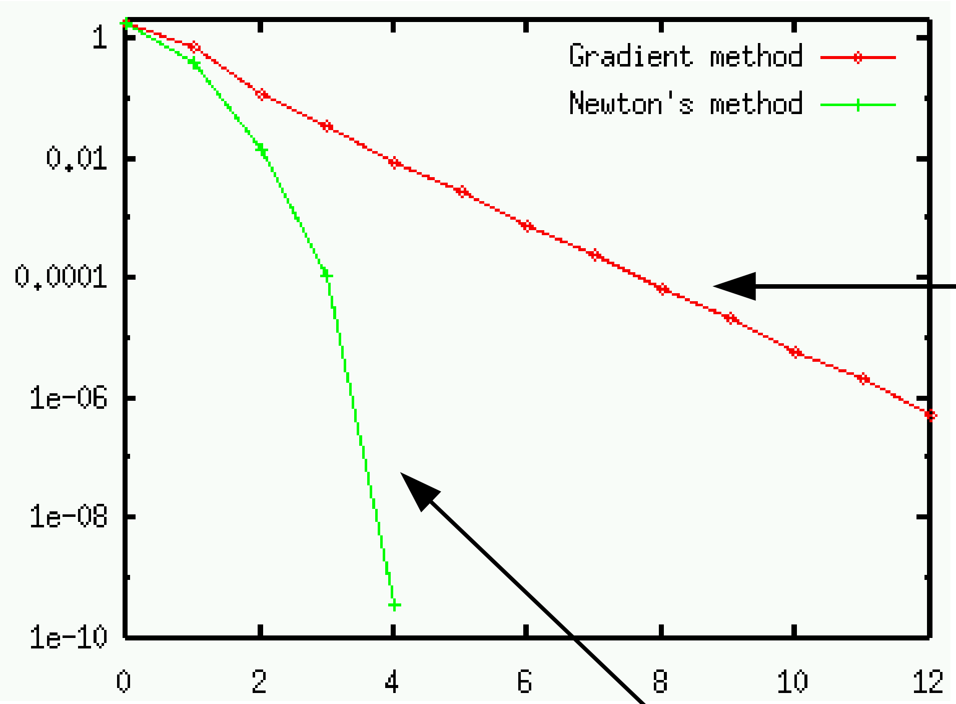
$$|a_1| \leq C|a_0|^2 \ \geq\ |a_0|$$

Generally, quadratic convergence is considered *fast* and we will want to use algorithms that have quadratic convergence.

How many iterations do we need?

Example: Compare linear and quadratic convergence

$\|x_k - x^*\|$



k

Linear convergence.

The gain factor C<1 is constant.

Quadratic convergence.

The 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

Different algorithms are best compared by comparing the number of function, gradient, or Hessian evaluations to achieve a certain accuracy, as this is a good measure for the run-time of such algorithms.

Part 4

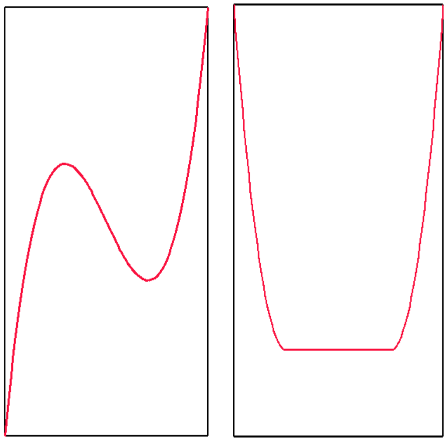
Smooth unconstrained problems: Line search algorithms

$$\text{minimize} \quad f(x)$$

Smooth problems: Characterization of Optima

Problem: find solution x^* of

$$\text{minimize}_x \ f(x)$$



A strict local minimum x^* must satisfy two conditions:

First order necessary condition: gradient must vanish:

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

Sufficient condition for a strict minimum:

$$\text{spectrum}(\nabla^2 f(x^*)) > 0$$

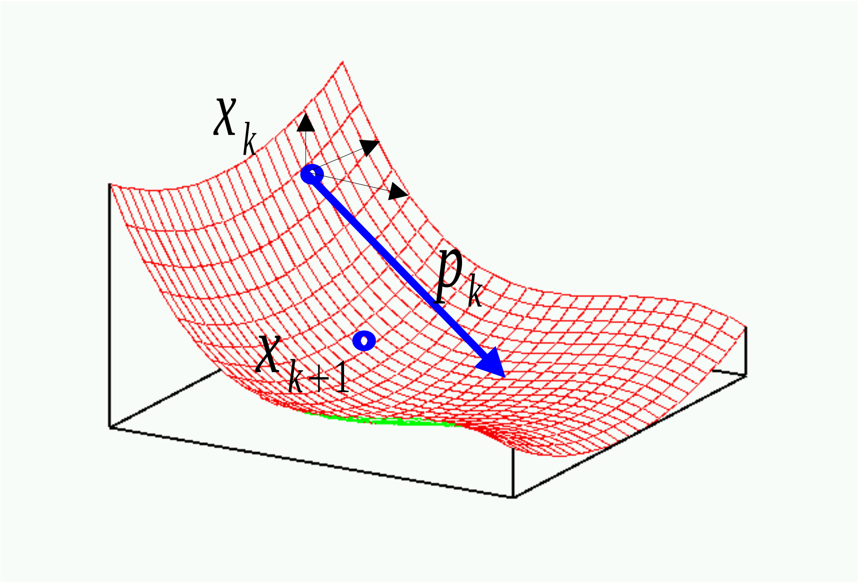
Basic Algorithm for Smooth Unconstrained Problems

Basic idea for iterative solution $x_k \rightarrow x^*$ of the minimization problem

$$\text{minimize} \quad f(x)$$

Generate a sequence x_k by

1. finding a search direction p_k
2. choosing a step length α_k



Then compute the update

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

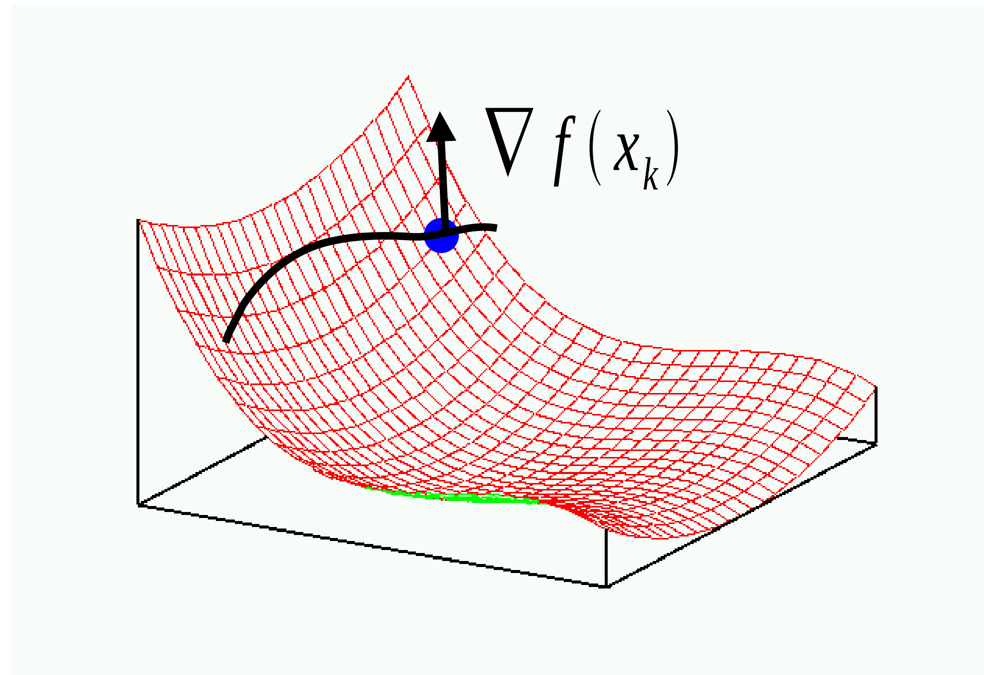
Iterate until we are satisfied.

Step 1: Choose search direction

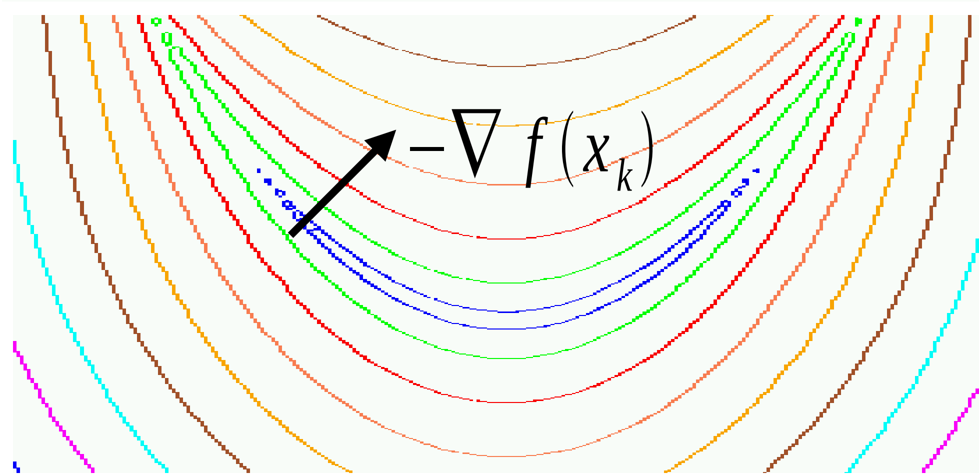
Conditions for a useful search direction:

Minimization function should be decreased in this direction:

$$p_k \cdot \nabla f(x_k) \leq 0$$



Search direction should lead to the minimum as straight as possible



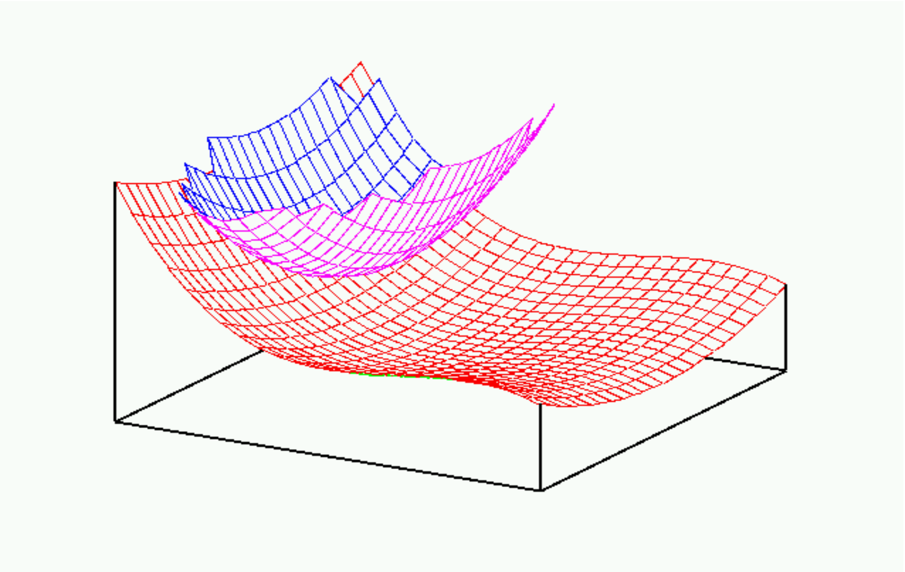
Step 1: Choose search direction

Basic assumption: we can usually only expect to know the minimization function $f(x_k)$ *locally* at x_k.
That means that we can only evaluate

$$f(x_k) \qquad \nabla f(x_k) = g_k \qquad \nabla^2 f(x_k) = H_k \qquad \ldots$$

For a search direction, try to model f in the vicinity of x_k by a Taylor series:

$$\begin{aligned} f(x_k + p_k) \approx\; & f(x_k) \\ & + g_k^T p_k \\ & + \frac{1}{2} p_k^T H_k p_k + \ldots \end{aligned}$$



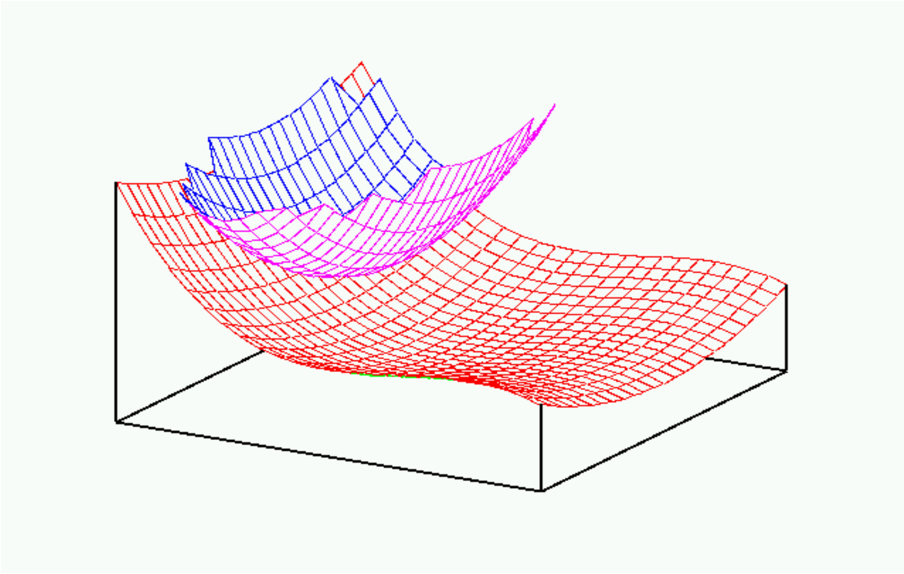
Step 1: Choose search direction

Goal: Approximate $f(\cdot)$ in the vicinity of x_k by a model

$$f(x_k+p) \approx m_k(p) = f_k + g_k^T p + \frac{1}{2} p^T H_k p + \ldots$$

with $\quad f(x_k)=f_k \qquad \nabla f(x_k)=g_k \qquad \nabla^2 f(x_k)=H_k \qquad \ldots$

Then: Choose that direction p_k that minimizes the model $m_k(p)$



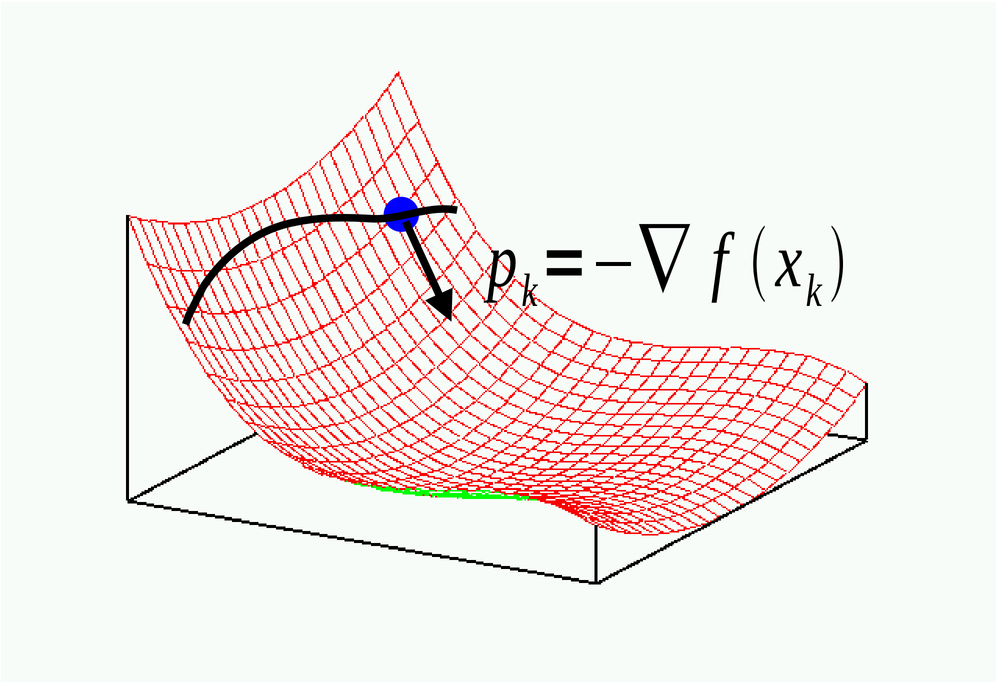
Step 1: Choose search direction

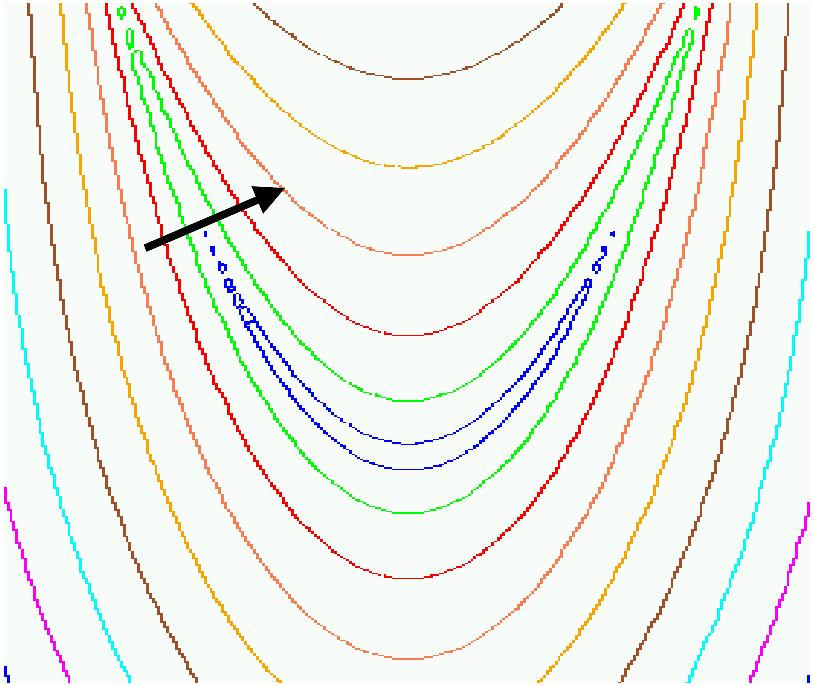
Method 1 (*Gradient method, Method of Steepest Descent*):

search direction is minimizing direction of *linear model*

$$f(x_k + p) \approx f_k + g_k^T p = m_k(p)$$

$$p_k = -g_k$$





Step 1: Choose search direction

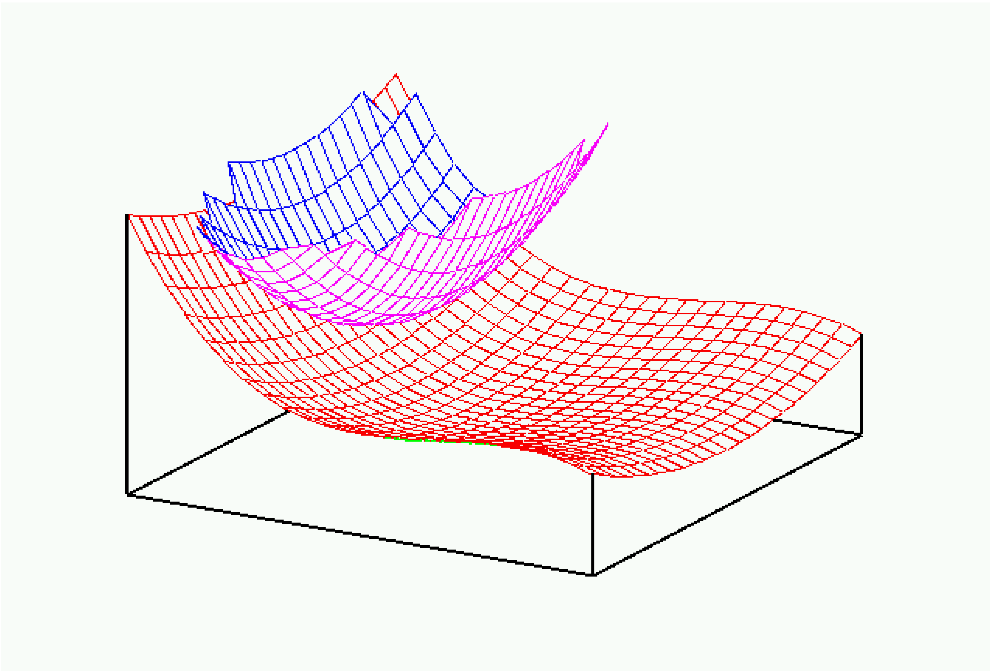
Method 2 (*Newton's method*):

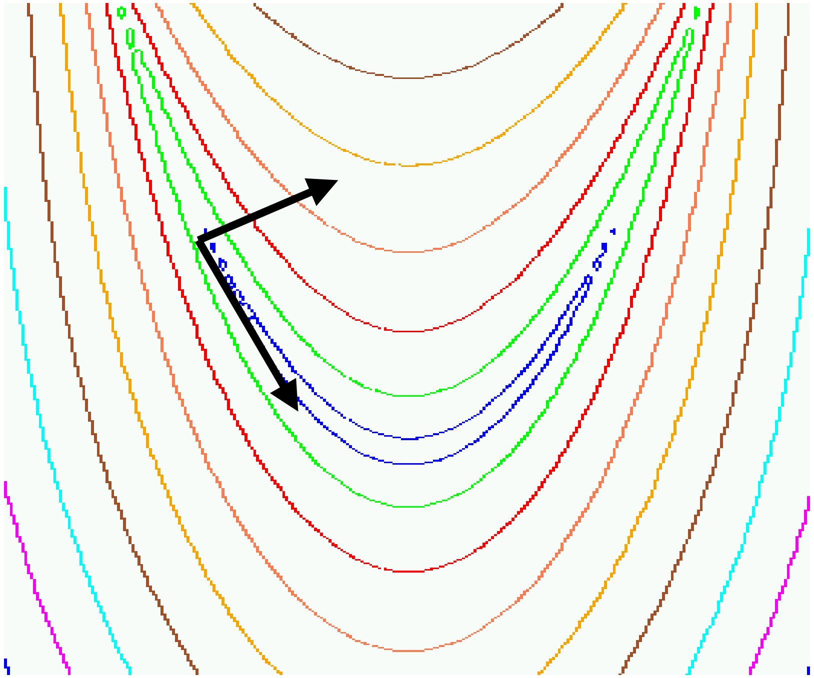
search direction is the direction to the minimum of the *quadratic model*

$$m_k(p) = f_k + g_k^T p + \frac{1}{2} p^T H_k p$$

Minimum is characterized by

$$\frac{\partial m_k(p)}{\partial p} = g_k + H_k p = 0 \quad \rightarrow \quad p_k = -H_k^{-1} g_k$$





Step 1: Choose search direction

Method 2 (*Newton's method*) -- alternative viewpoint:

Newton step is also generated when applying Newton's method for the root-finding problem ($F(x)=0$) to the necessary optimality condition:

Linearize necessary condition around x_k:

$$0 = \nabla f(x^*) = \underbrace{\nabla f(x_k)}_{g_k} + \underbrace{\nabla^2 f(x_k)}_{H_k} \underbrace{(x^* - x_k)}_{p_k} + \ldots$$

$$p_k = -H_k^{-1} g_k$$

Step 1: Choose search direction

Method 3 (*A third order method*):

The search direction is the direction to the minimum of the *cubic model*

$$m_k(p) = f_k + g_k^T p + \frac{1}{2} p^T H_k p + \frac{1}{6}\left[\frac{\partial^3 f}{\partial x_l \partial x_m \partial x_n}\right]_k p_l p_m p_n$$

The minimum of this model is characterized by the quadratic equation

$$\frac{\partial m_k(p)}{\partial p} = g_k + H_k p + \frac{1}{2}\left[\frac{\partial^3 f}{\partial x_l \partial x_m \partial x_n}\right]_k p_l p_m = 0 \quad \rightarrow \quad p_k = ???$$

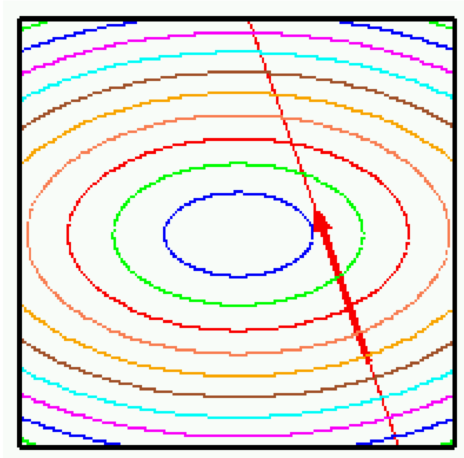
There doesn't appear to be any practical way to compute the solution of this equation for problems with more than one variable.

Step 2: Determination of Step Length

Once the search direction is known, compute the update by choosing a step length α_k and set

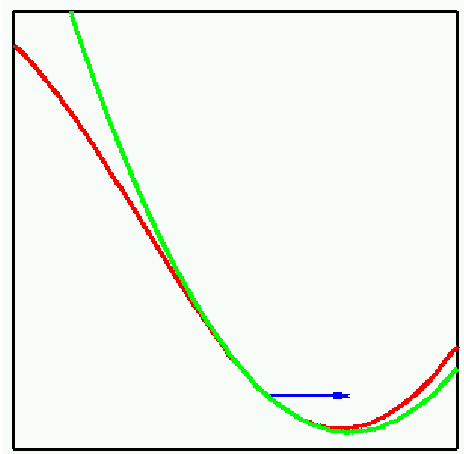
$$x_{k+1} = x_k + \alpha_k p_k$$

Determine the step length by solving the 1-d minimization problem (*line search*):



$$\alpha_k = \arg\min_\alpha \; f(x_k + \alpha p_k)$$

For Newton's method: If the quadratic model is good, then step is good, then take *full step* with $\alpha_k = 1$



Convergence: Gradient method

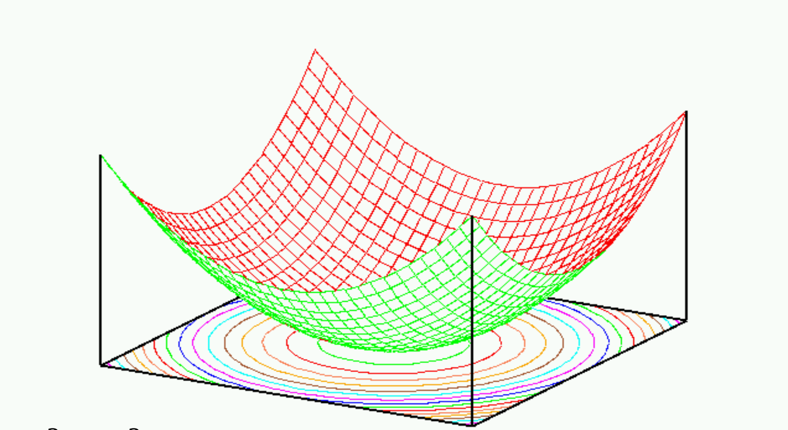
Gradient method converges *linearly*, i.e.

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

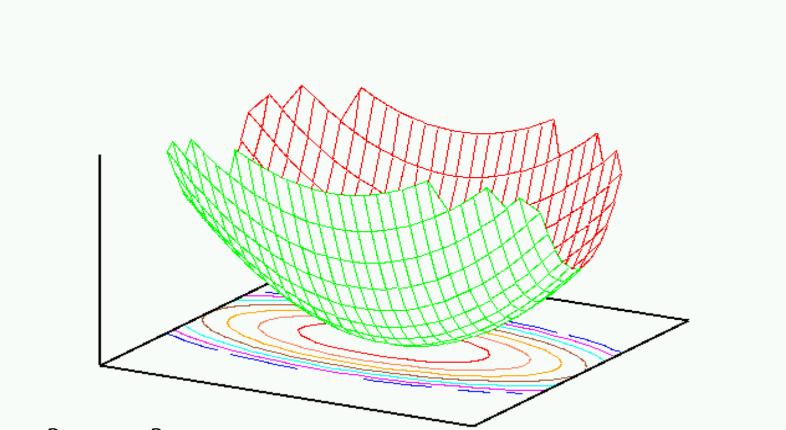
Gain is a fixed factor $C<1$
Convergence can be *very* slow if C close to 1.

Example: If $f(x)=x^T H x$, with H positive definite and for optimal line search, then

$$C \approx \frac{\lambda_n - \lambda_1}{\lambda_n + \lambda_1} \qquad \{\lambda_i\} = \text{spectrum } H$$



$x^2 + y^2 \;\rightarrow\; C = 0$



$x^2 + 5y^2 \;\rightarrow\; C \approx 0.6$

Convergence: Newton's method

Newton's method converges *quadratically*, i.e.

$$\|x_k - x^*\| \;\leq\; C\|x_{k-1} - x^*\|^2$$

Optimal convergence order only if step length is 1, otherwise slower convergence (step length is 1 if quadratic model valid!)

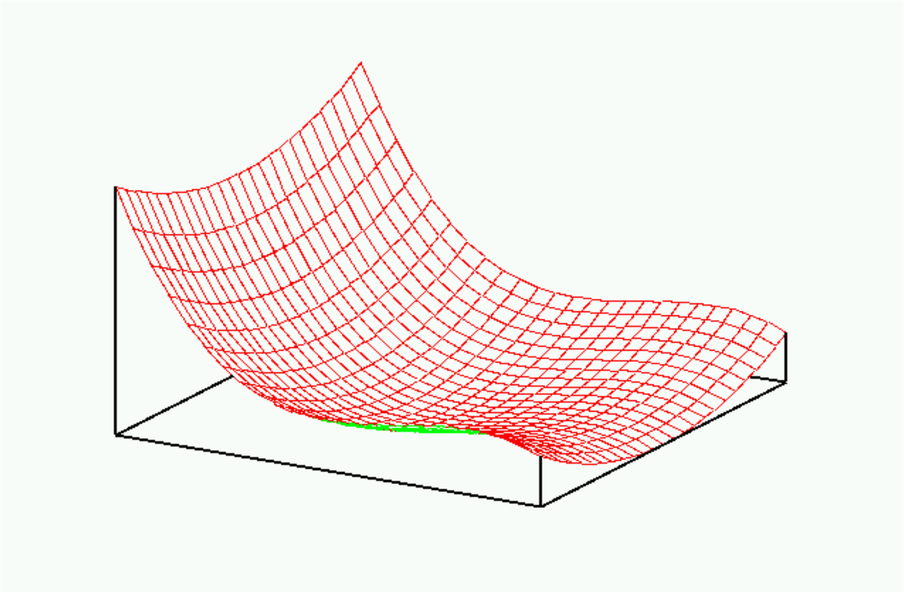
If quadratic convergence: accelerating progress as iterations proceed.

Size of C:

$$C \;\sim\; \frac{\left\|\nabla^2 f(x^*)^{-1}\left(\nabla^2 f(x) - \nabla^2 f(y)\right)\right\|}{\|x - y\|}$$

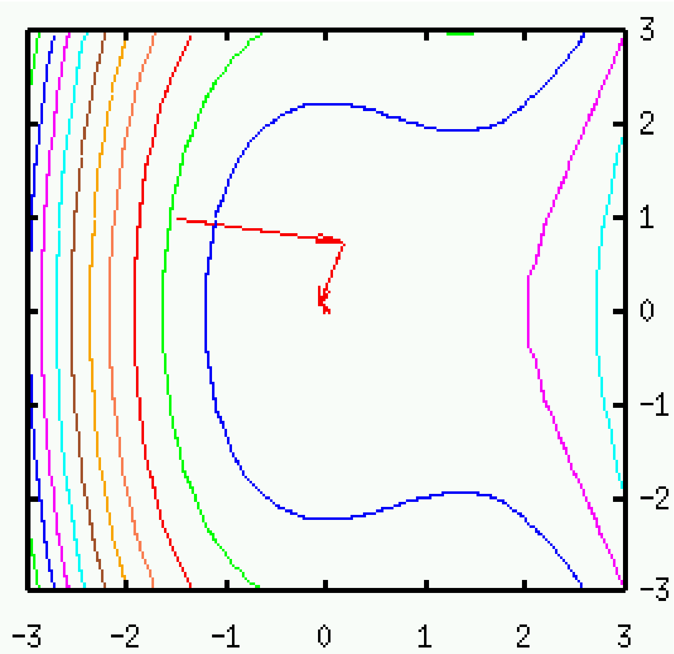
C measures size of nonlinearity beyond quadratic part.

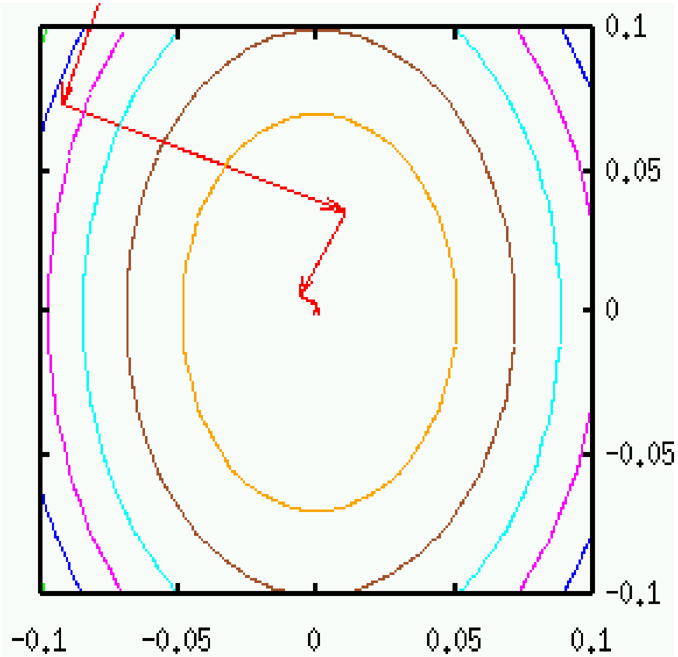
Example 1: Gradient method



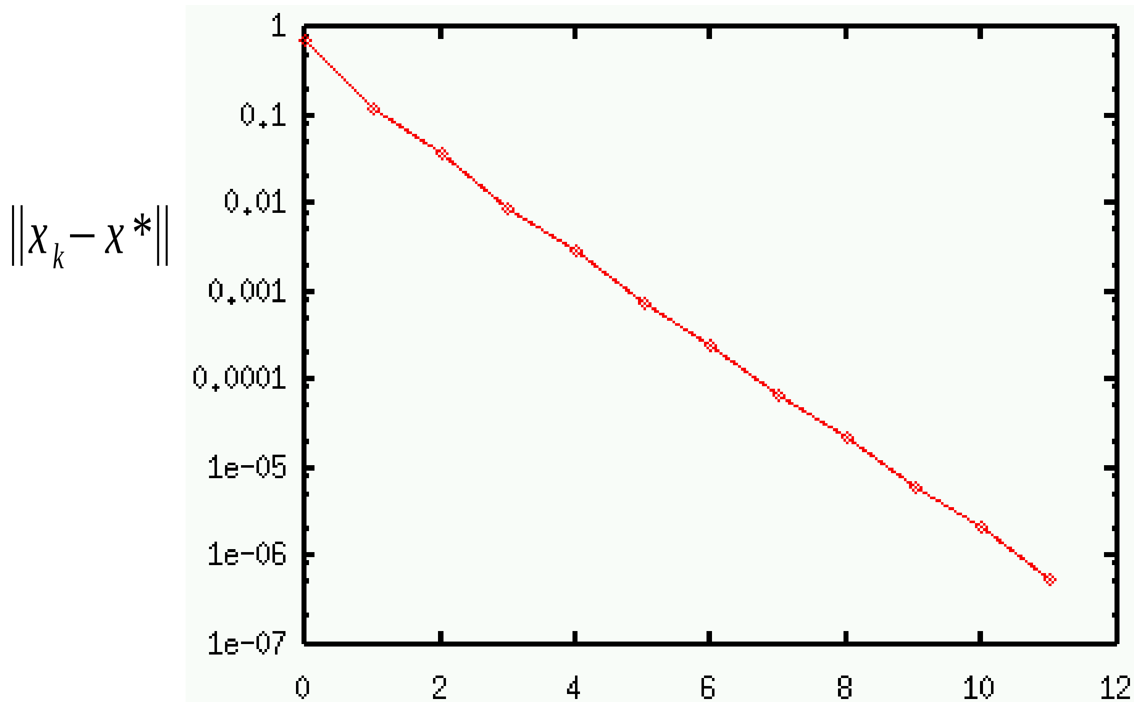
$$f(x,y) = -x^3 + 2x^2 + y^2$$

Local minimum at x=y=0,
saddle point at x=4/3, y=0





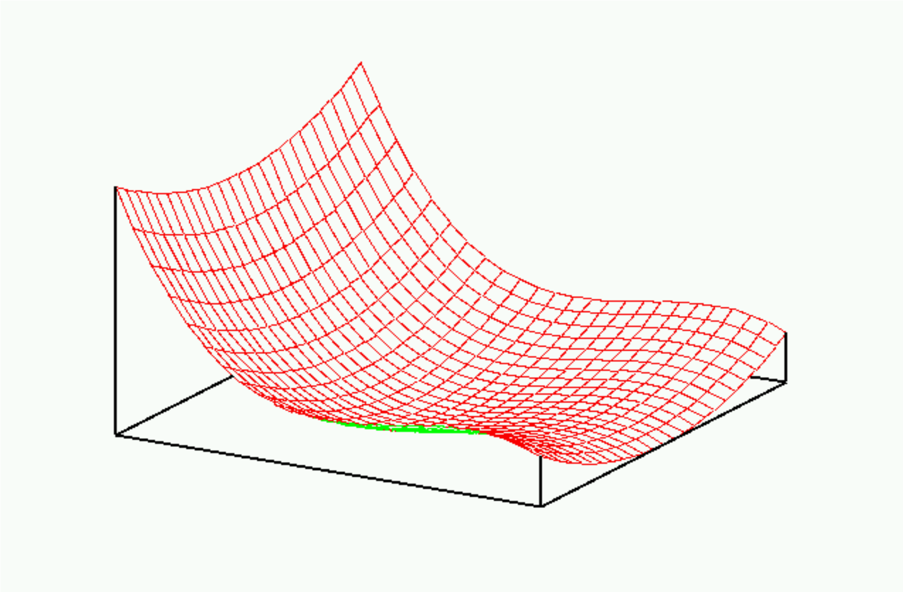
<u>Example 1: Gradient method</u>



Convergence of gradient method:
Converges quite fast, with *linear* rate
Mean value of convergence constant C : 0.28
At (x=0,y=0), there holds

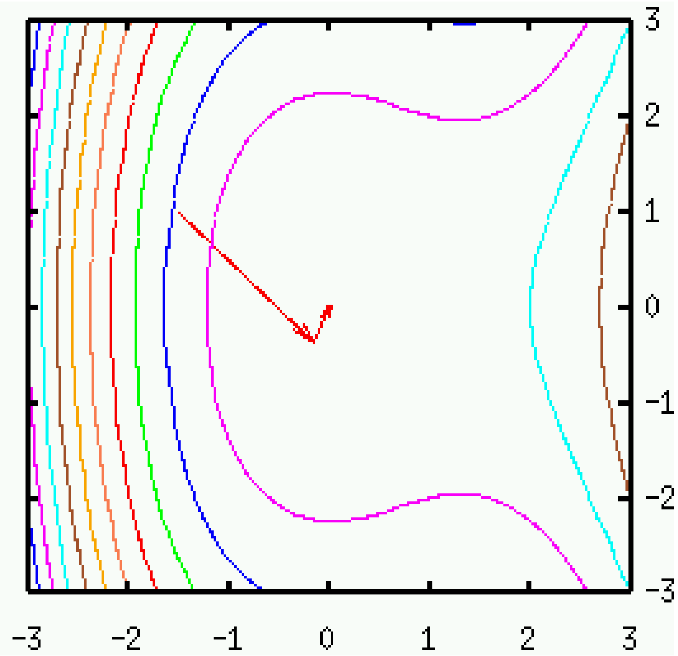
$$\nabla^2 f(0,0) \sim \{\lambda_1 = 4, \lambda_2 = 2\} \qquad C \approx \frac{4-2}{4+2} \approx 0.33$$

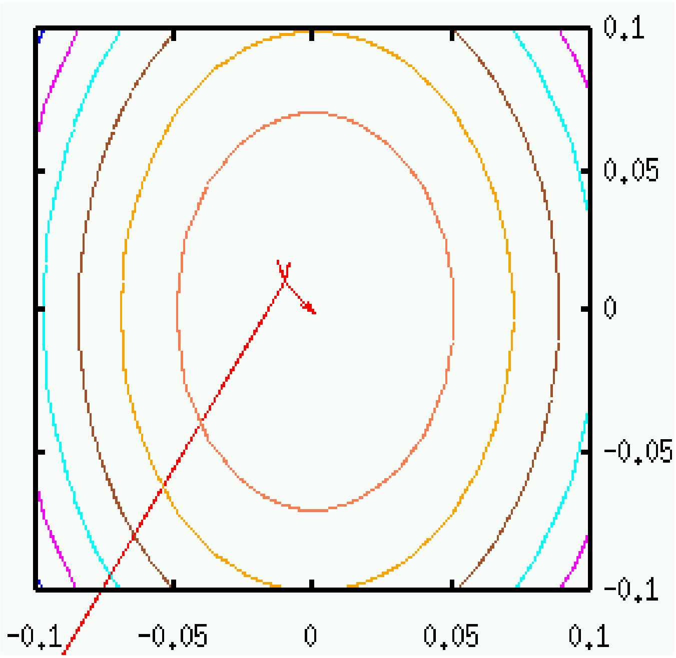
Example 1: Newton's method



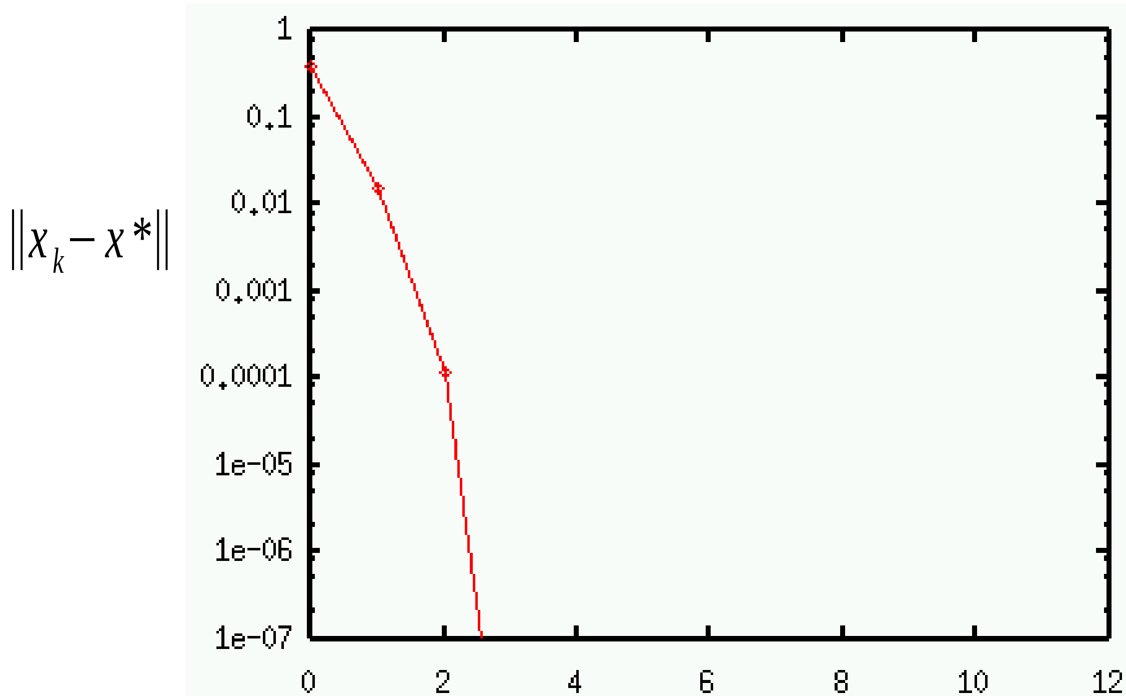
$$f(x,y) = -x^3 + 2x^2 + y^2$$

Local minimum at x=y=0, saddle point at x=4/3, y=0





Example 1: Newton's method



Convergence of Newton's method:

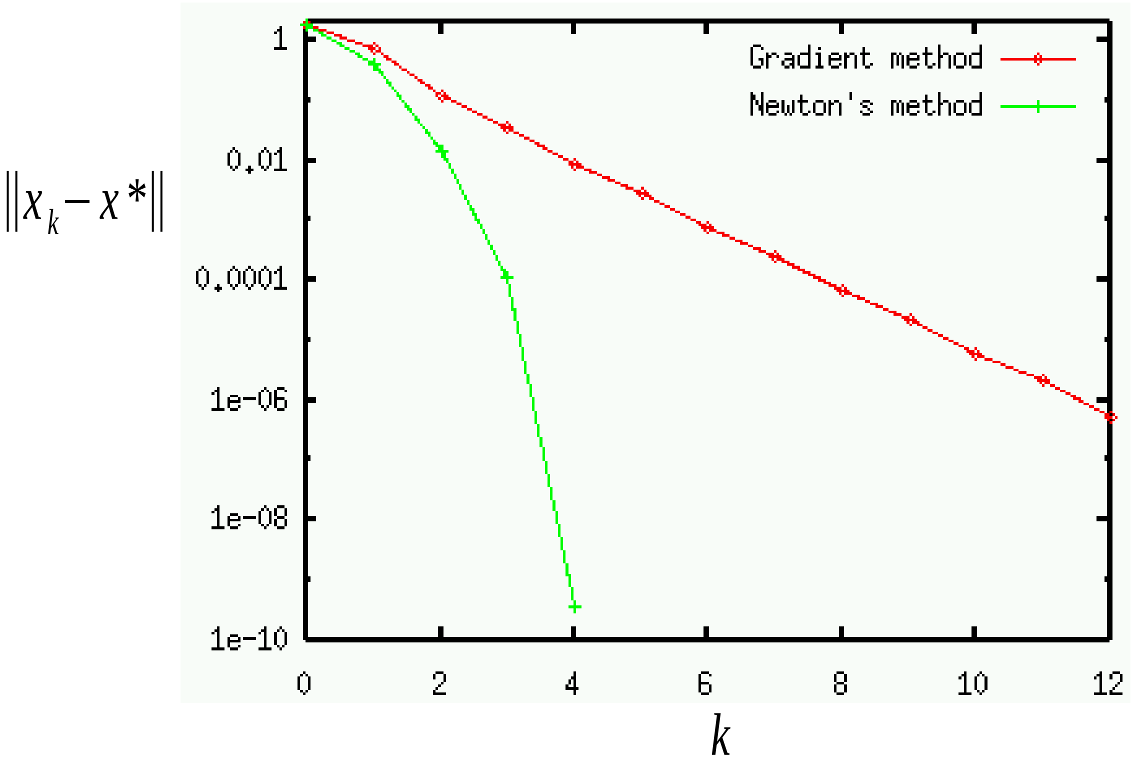
Converges very fast, with *quadratic* rate

Mean value of convergence constant C : 0.15

$$\|x_k - x^*\| \le C\,\|x_{k-1} - x^*\|^2$$

Theoretical estimate yields C=0.5

Example 1: Comparison between methods

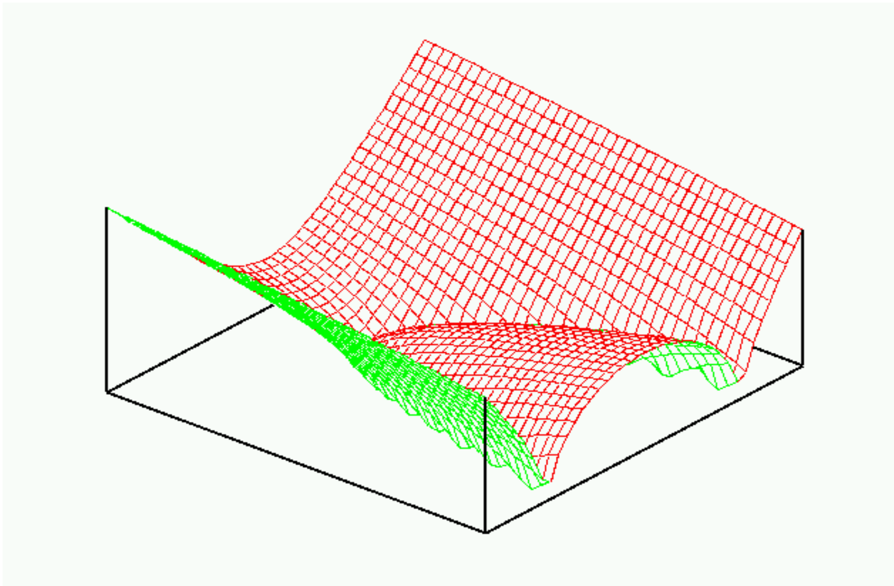


Newton's method much faster than gradient method

Newton's method superior for high accuracy due to higher order of convergence

Gradient method simple but converges in a reasonable number of iterations as well

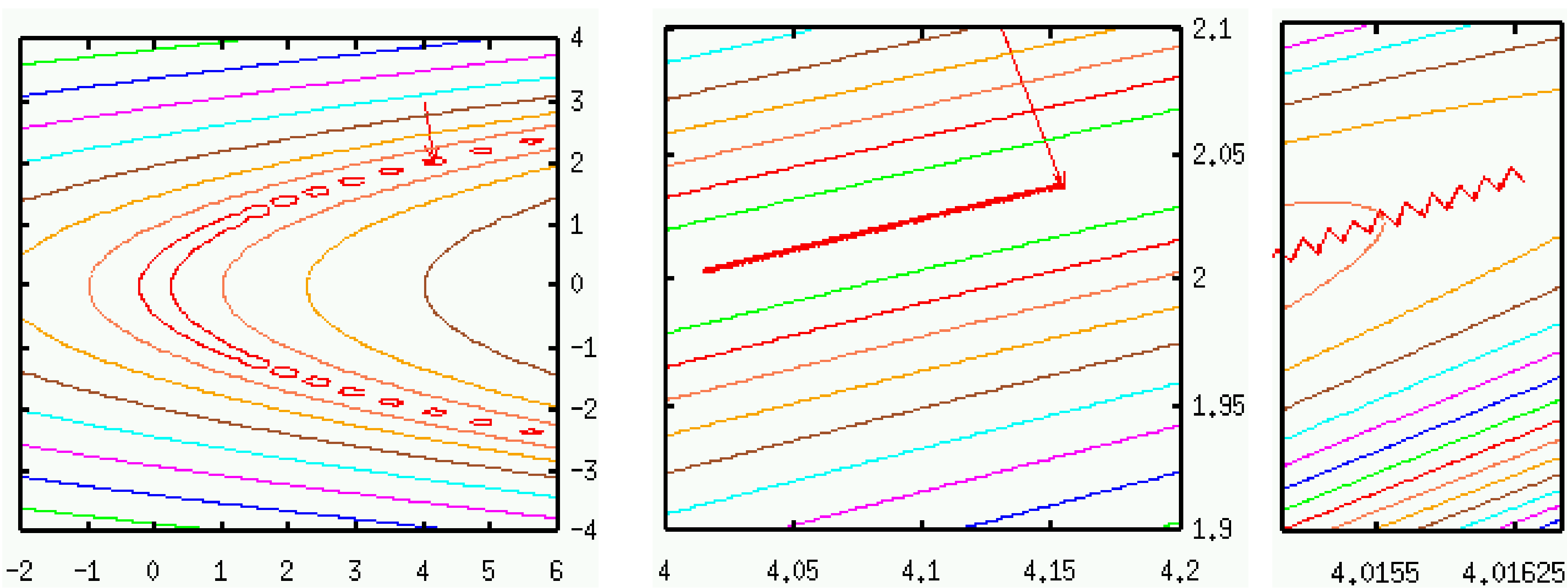
Example 2: Gradient method



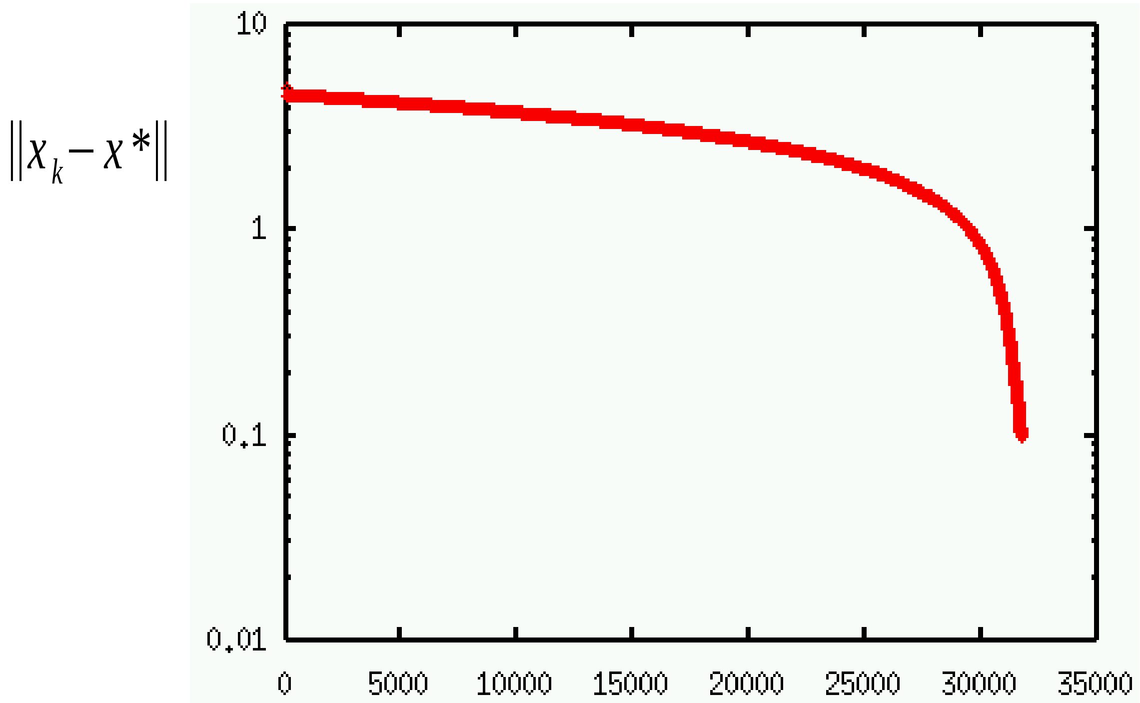
$$f(x,y)=\sqrt[4]{\left((x-y^2)^2+\frac{1}{100}\right)}+\frac{1}{100}y^2$$

(*Banana valley* function)

Global minimum at x=y=0



Example 2: Gradient method



Convergence of gradient method:

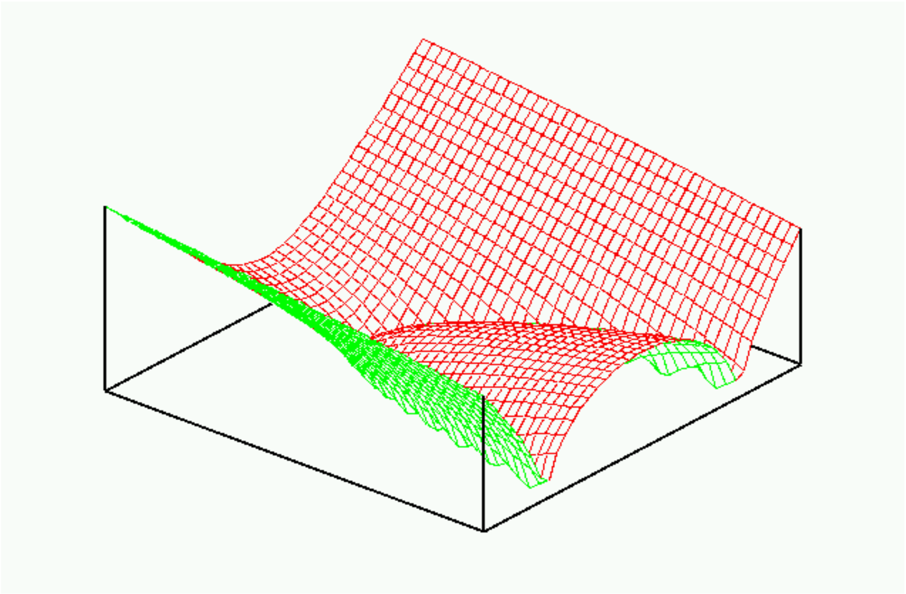
Needs almost 35.000 iterations to come closer than 0.1 to the solution!

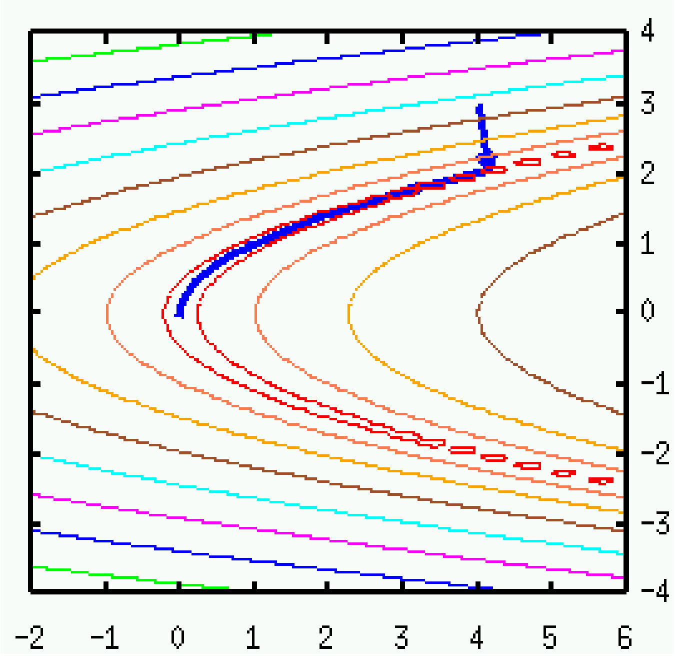
Mean value of convergence constant C : 0.99995

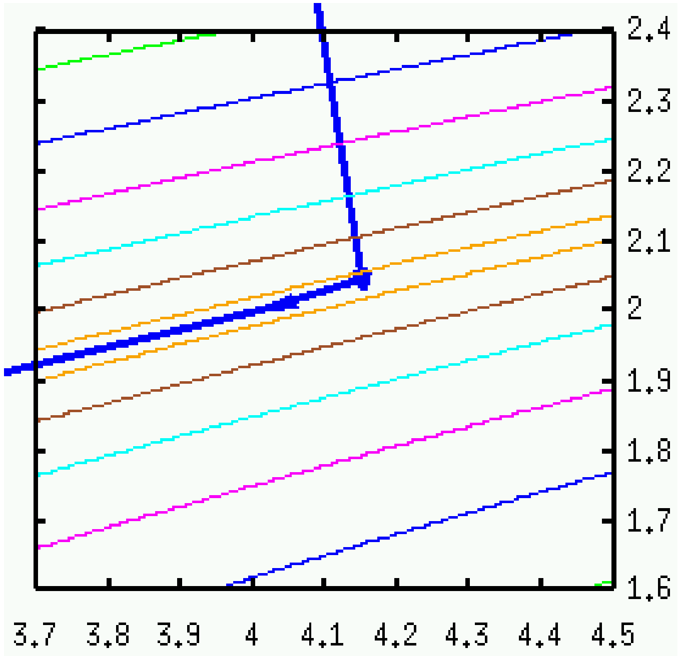
At (x=4,y=2), there holds

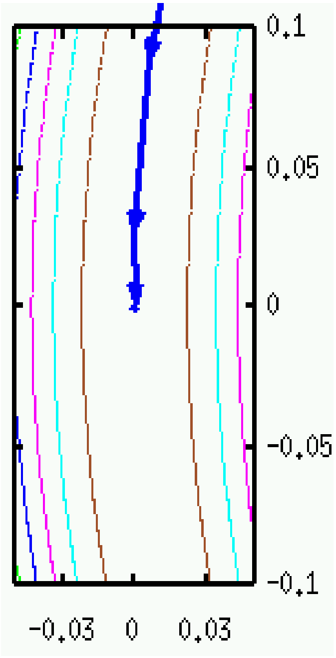
$$\nabla^2 f(4,2) \sim \{\lambda_1 = 0.1, \lambda_2 = 268\} \qquad C \approx \frac{268 - 0.1}{268 + 0.01} \approx 0.9993$$

Example 2: Newton's method

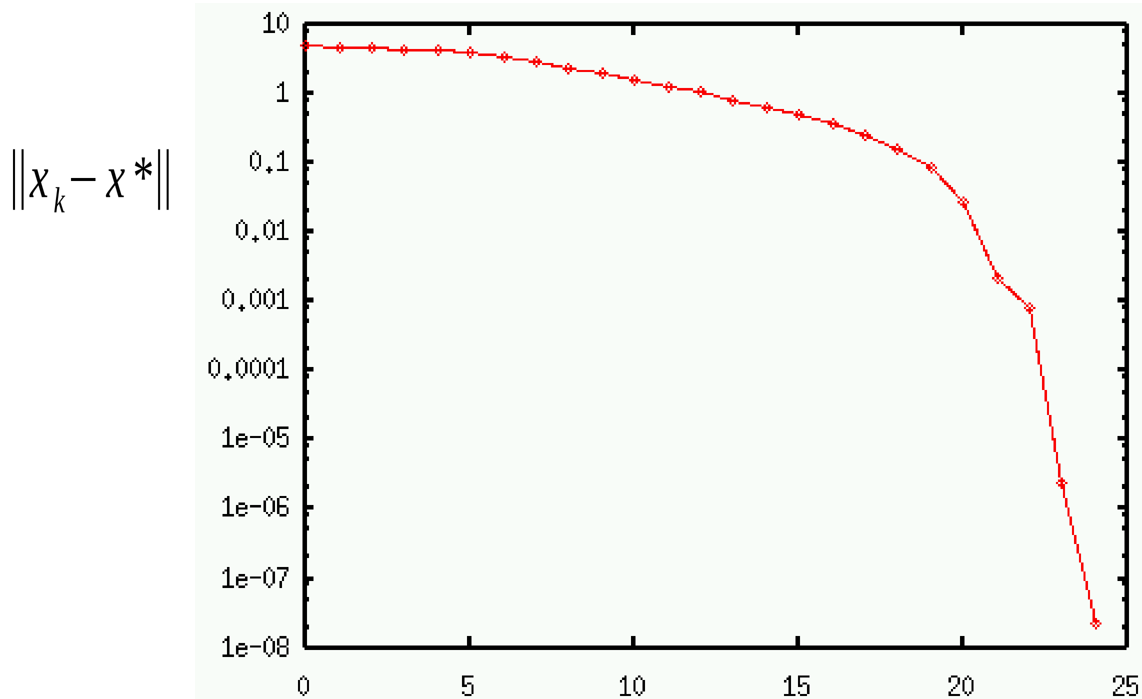








Example 2: Newton's method



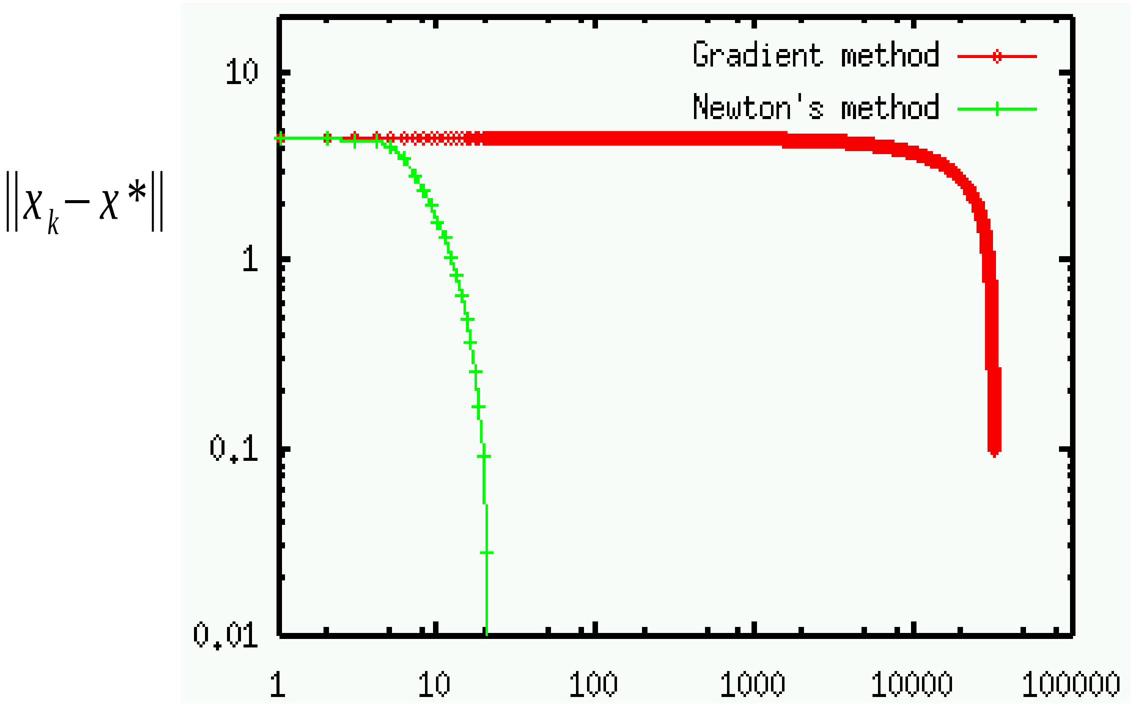
Convergence of Newton's method:

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 $\alpha_k \approx 1$

Example 2: Comparison between methods



Newton's method much faster than gradient method

Newton's method superior for high accuracy (i.e. in the vicinity of the solution) due to higher order of convergence

Gradient method converges too slow for practical application

Practical line search strategies

Ideally: We would use an exact step length determination (*line search*) based on

$$\alpha_k = \arg\min_\alpha f(x_k + \alpha p_k)$$

This is a one-dimensional minimization problem for α, which one could implement using Newton's method or bisection search or predictor-corrector methods, or ....

However: This is expensive, since it may require many function or gradient evaluations.

Instead: Find practical criteria that guarantee convergence but need less function evaluations!

Practical line search strategies

Strategy: Find practical criteria that guarantee convergence but need less evaluations.

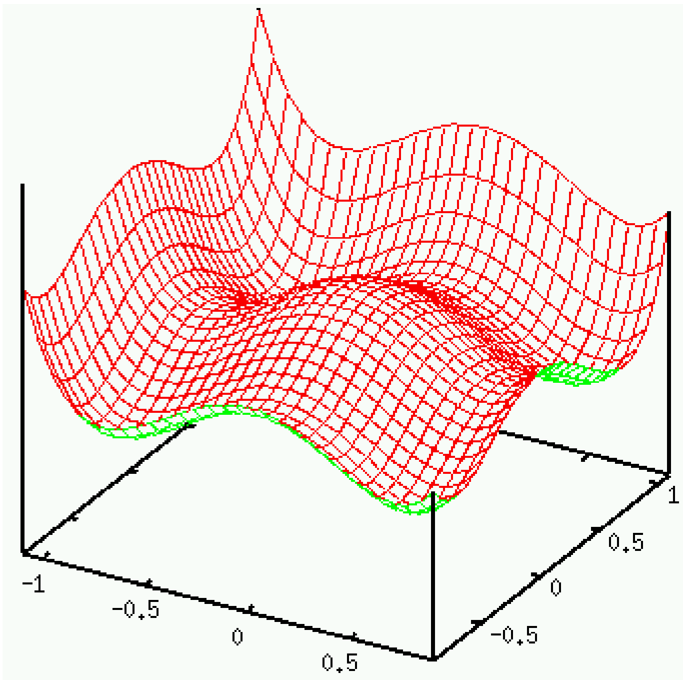
Rationale:

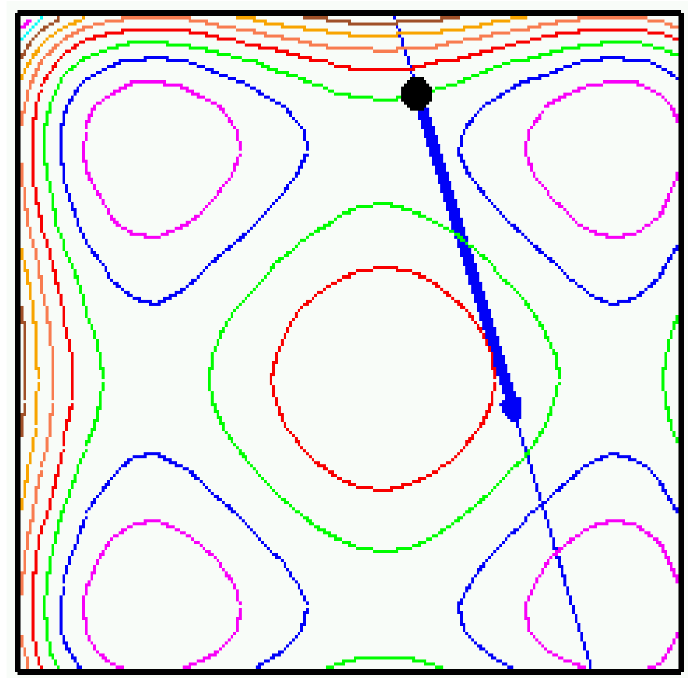
- Near the optimum, the quadratic approximation of f is valid, so we may take full steps (step length 1) there
- Inexact line search only necessary when far away from the solution
- If close to solution, our strategy will try α=1 first
- Quadratic convergence of Newton's method therefore retained near the solution; when far away, convergence is slower in any case.

Practical line search strategies

Practical strategy: Replace exact line search by a strategy that

- finds a reasonable approximation to the exact step length
- chosen step length guarantees a *sufficient decrease* in *f(x)*;
- chooses full step length 1 for Newton's method whenever possible.





$$f(x, y) = x^4 - x^2 + y^4 - y^2$$

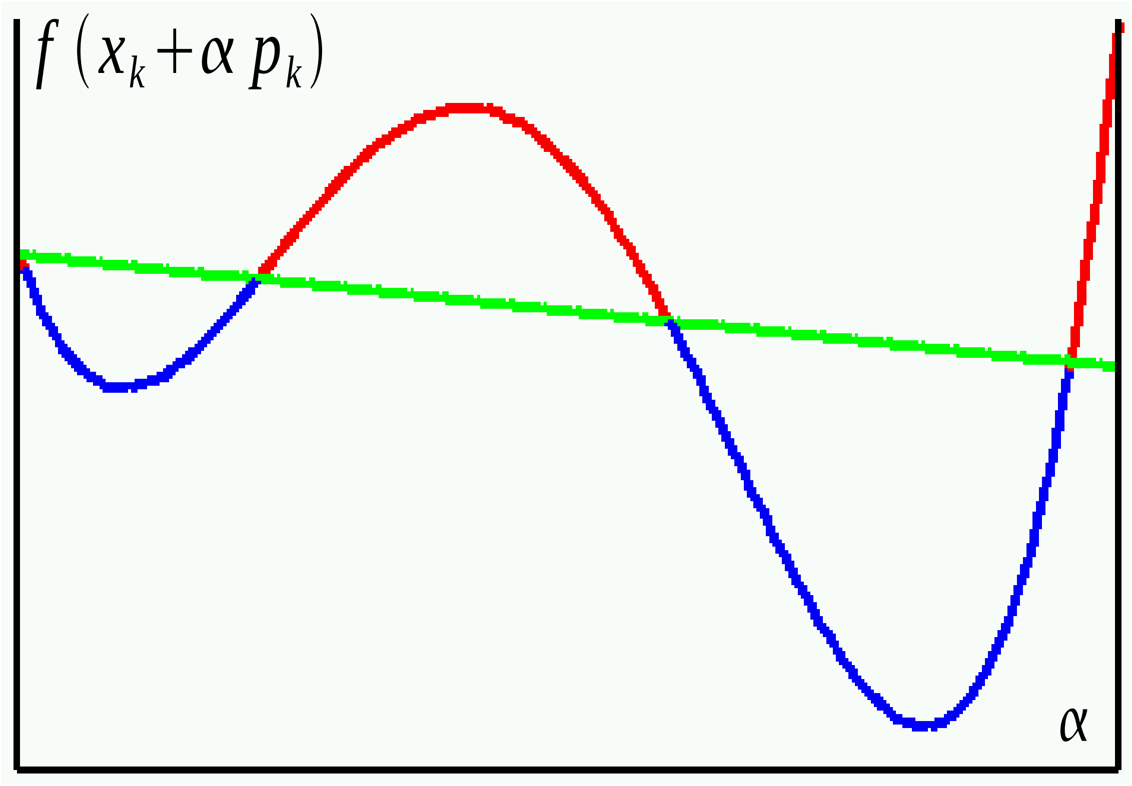
Practical line search strategies

Wolfe condition 1:

Only allow those step lengths that produce a sufficient decrease

$$f(x_k+\alpha p_k) \;\leq\; f(x_k) + c_1\alpha\left[\frac{\partial f(x_k+\alpha p_k)}{\partial\alpha}\right]_{\alpha=0}$$

$$= f_k + c_1\alpha\nabla f_k\cdot p_k$$



Necessary:

$$0<c_1<1$$

Typical values:

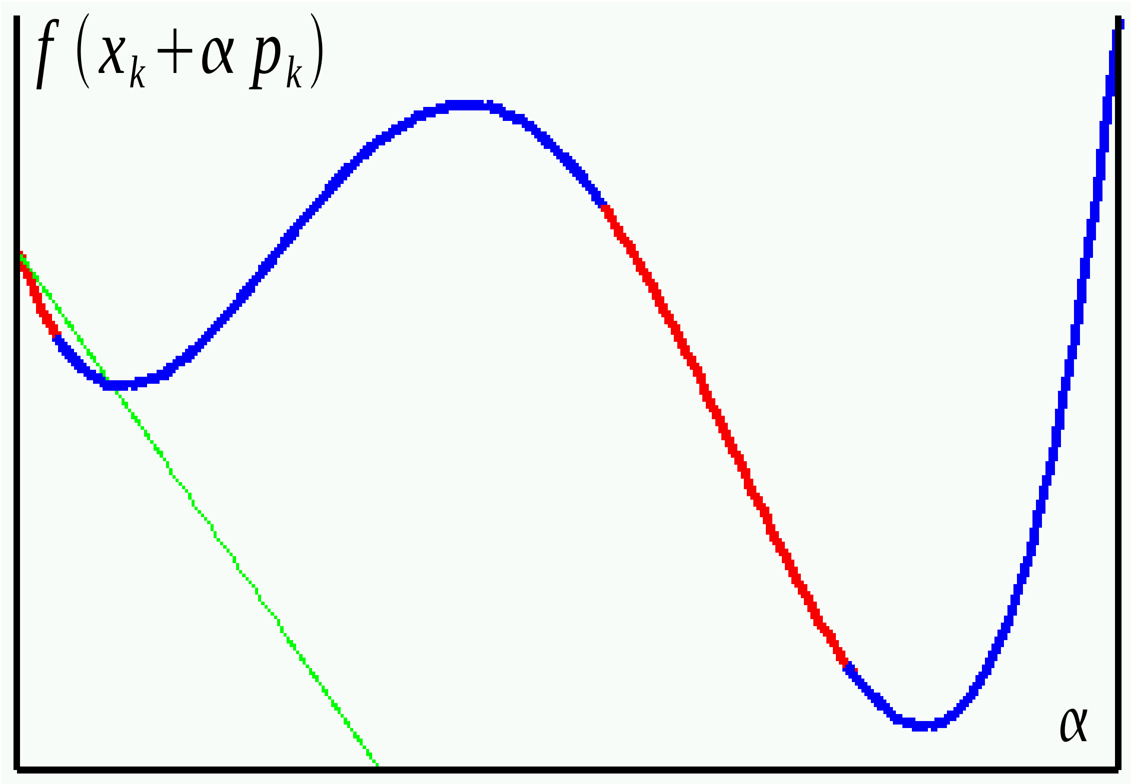
$$c_1=10^{-4}$$

i.e.: only very small decrease mandated

Practical line search strategies

Wolfe condition 2 ("curvature condition"):
Only allow those step lengths at which *f* has shown sufficient curvature upwards

$$\nabla f(x_k+\alpha p_k)\cdot p_k = \left[\frac{\partial f(x_k+\alpha p_k)}{\partial \alpha}\right]_{\alpha=\alpha_k} \geq c_2\left[\frac{\partial f(x_k+\alpha p_k)}{\partial \alpha}\right]_{\alpha=0} = c_2\nabla f_k\cdot p_k$$



Necessary:

$$0<c_1<c_2<1$$

Typical:

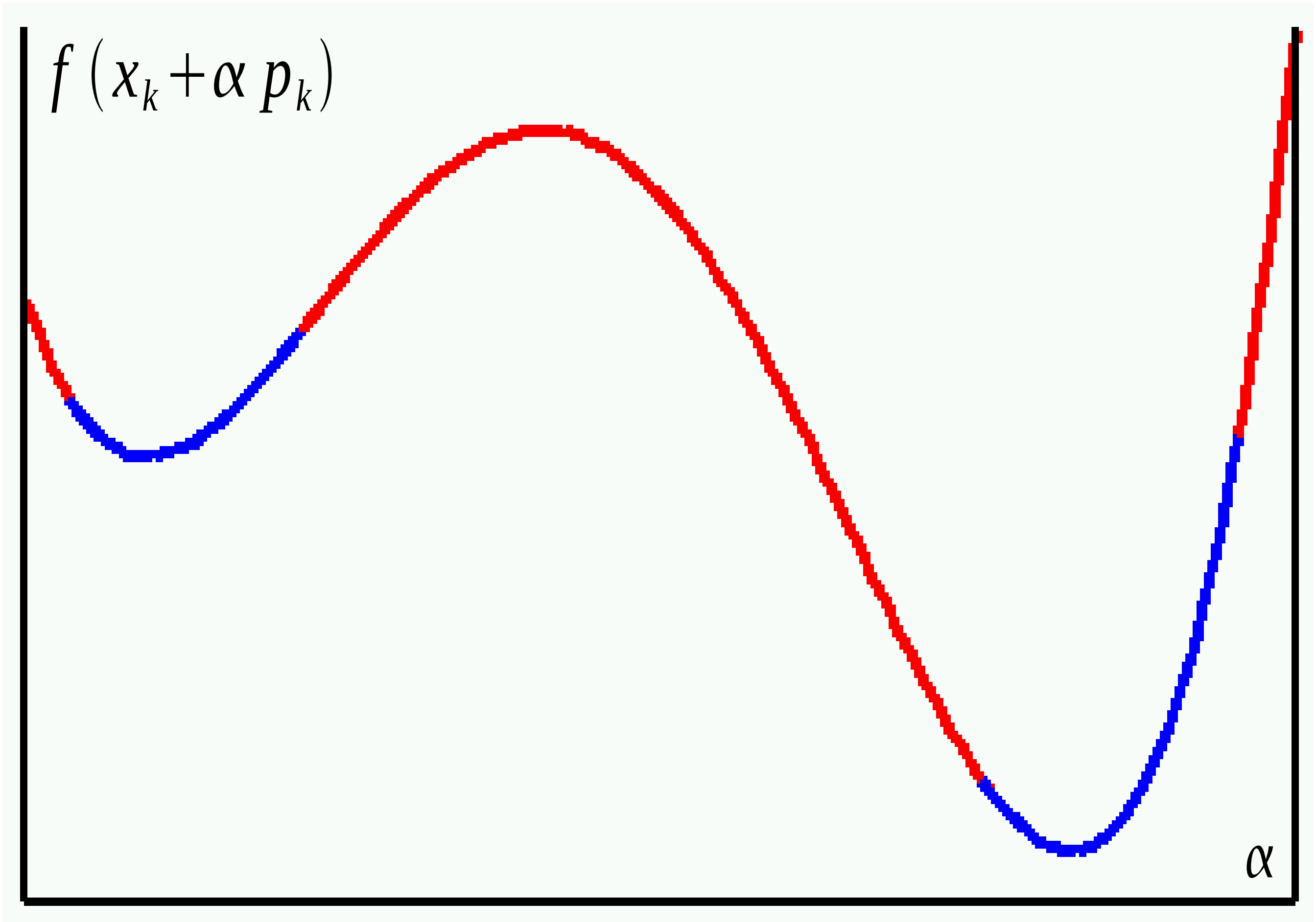
$$c_2=0.9$$

Rationale: Exclude too small step lengths

Practical line search strategies

Wolfe conditions

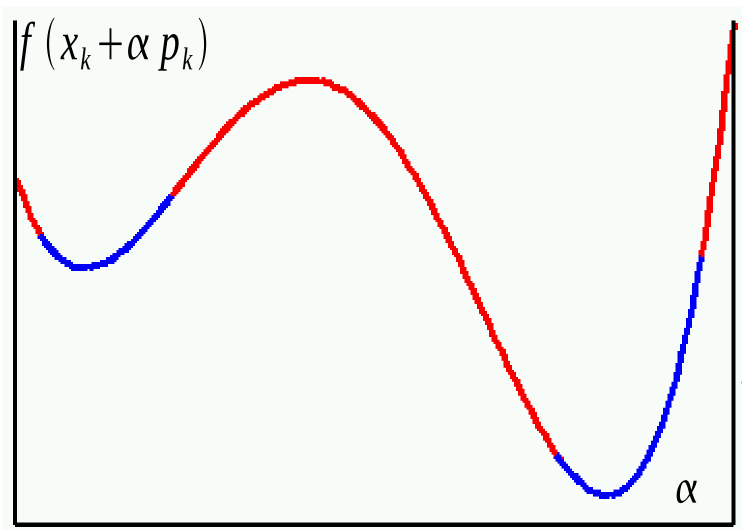
Conditions 1 and 2 usually yield reasonable ranges for the step lengths, but do not guarantee optimal ones

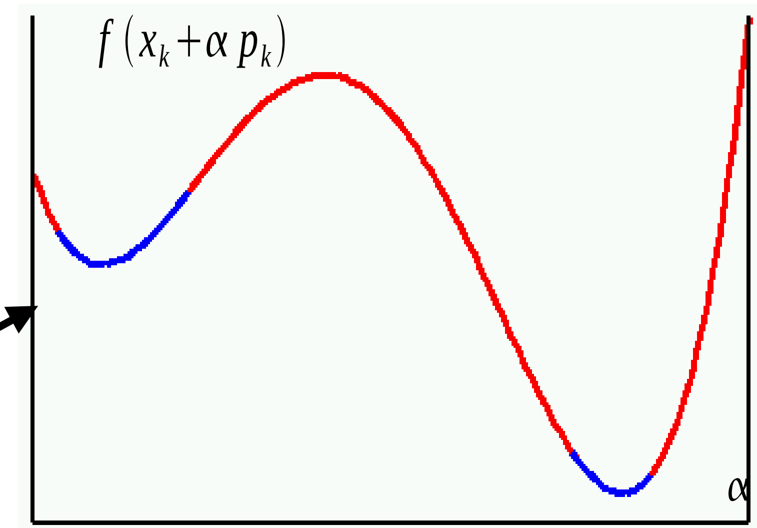


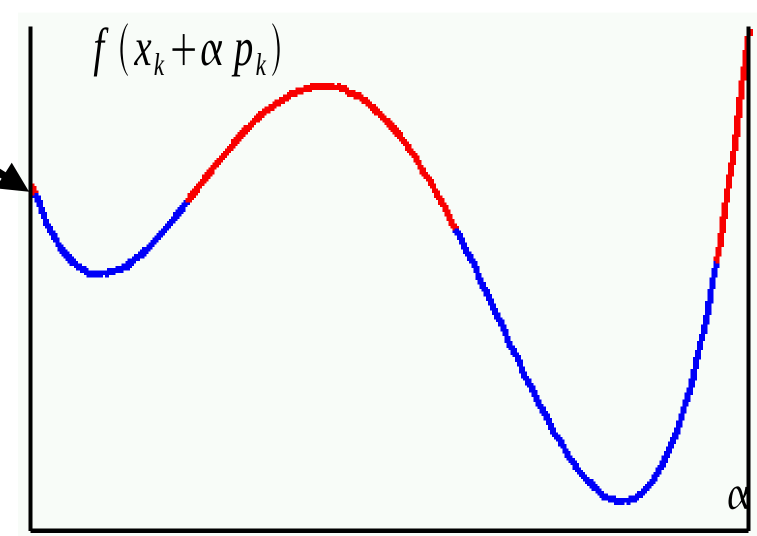
Practical line search strategies - Alternatives

Strict Wolfe conditions:

$$\left|\left[\frac{\partial f(x_k+\alpha p_k)}{\partial \alpha}\right]_{\alpha=\alpha_k}\right| \le c_2\left|\left[\frac{\partial f(x_k+\alpha p_k)}{\partial \alpha}\right]_{\alpha=0}\right|$$







Goldstein conditions:

$$f(x_k+\alpha p_k) \ge f(x_k) + (1-c_1)\alpha\left[\frac{\partial f(x_k+\alpha p_k)}{\partial \alpha}\right]_{\alpha=0}$$

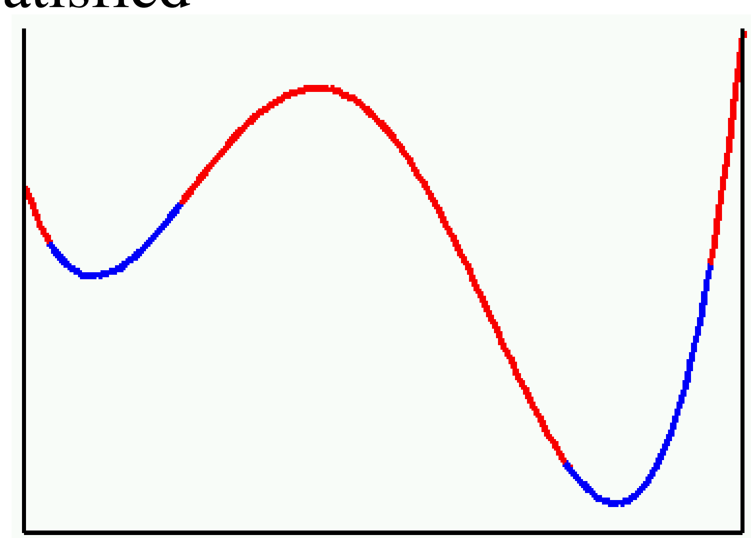
Practical line search strategies

Conditions like the Wolfe conditions tell us whether a given step length is acceptable or not.

However, in practice we don't want to try too many step lengths since checking the conditions involves function evaluations of *f(x)*.

Typical strategy ("Backtracking line search"):

- Start with a trial step length $\alpha_t = \bar{\alpha}$ (for Newton's method, we want full step length so $\bar{\alpha} = 1$)
- Verify if for this α_t the conditions are satisfied
- If so, then choose $\alpha_k = \alpha_t$
- If not, set $\alpha_t = c\,\alpha_t,\ c < 1$ and try step 2 again
- A typical reduction factor is $c = \frac{1}{2}$



Practical line search strategies

An alternative strategy ("Interpolating line search"):

- Start with a trial step length $\alpha_t^{(0)} = \bar{\alpha}$, set $i = 0$
  (for Newton's method, we want full step length so $\bar{\alpha} = 1$)
- Verify if for this $\alpha_t^{(i)}$ the Wolfe conditions are satisfied
- If so, then choose $\alpha_k = \alpha_t^{(i)}$
- If not:
  - let $\phi_k(\alpha) = f(x_k + \alpha p_k)$
  - in order to evaluate the sufficient decrease condition

$$f(x_k + \alpha_t^{(i)} p_k) \leq f_k + c_1 \alpha_t^{(i)} \nabla f_k \cdot p_k$$

  we already had to evaluate $\phi_k(0) = f(x_k)$, $\phi_k'(0) = \nabla f_k \cdot p_k = g_k \cdot p_k$
  and $\phi_k(\alpha_t^{(i)}) = f(x_k + \alpha_t^{(i)} p_k)$

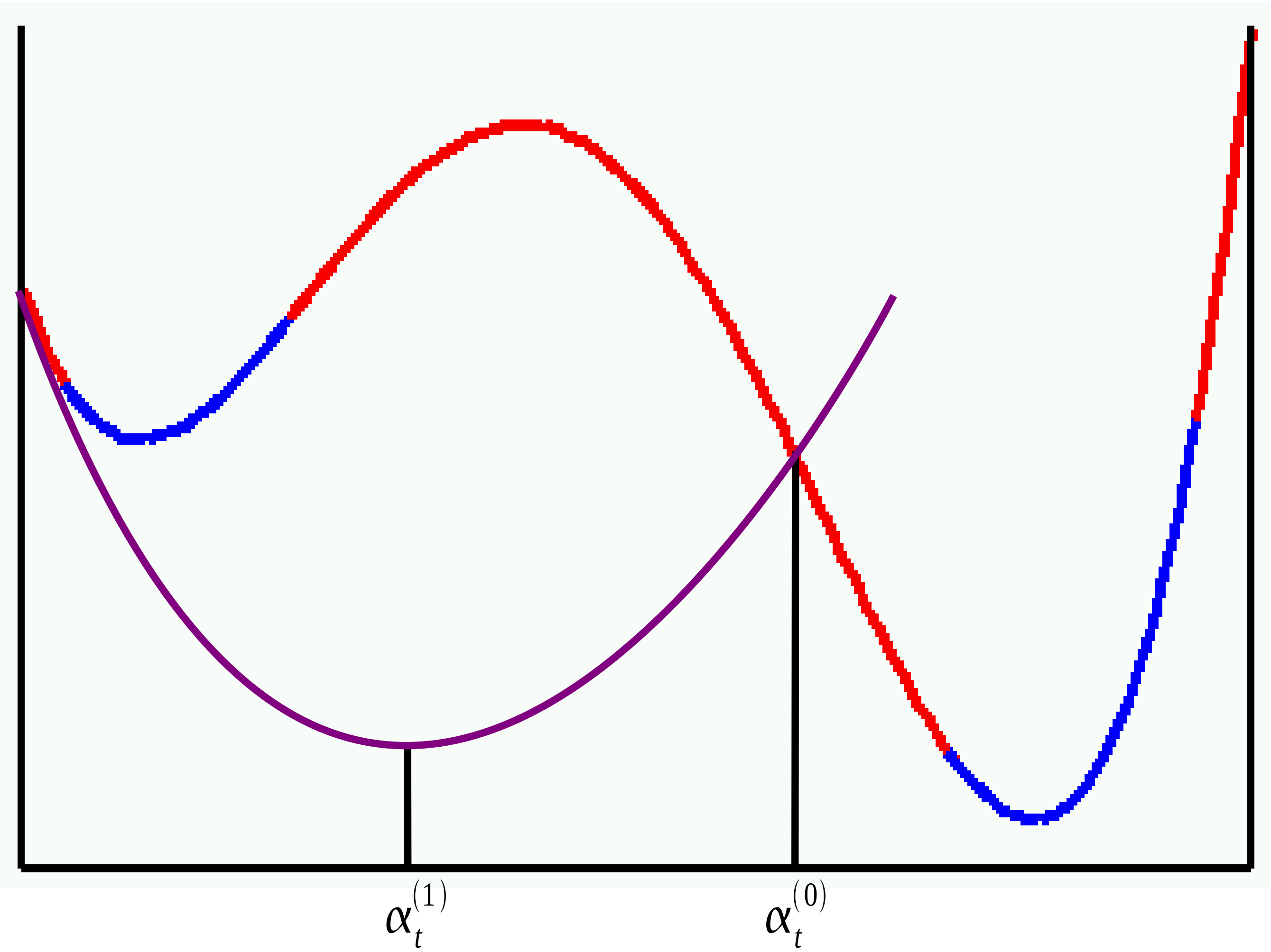
  - if $i = 0$ then choose $\alpha_t^{(i+1)}$ as the minimizer of the quadratic function that interpolates $\phi_k(0), \phi'_k(0), \phi_k(\alpha_t^{(i)})$

  - if $i > 0$ then choose $\alpha_t^{(i+1)}$ as the minimizer of the cubic function that interpolates $\phi_k(0), \phi'_k(0), \phi_k(\alpha_t^{(i)}), \phi_k(\alpha_t^{(i-1)})$

Practical line search strategies

An alternative strategy ("Interpolating line search"):

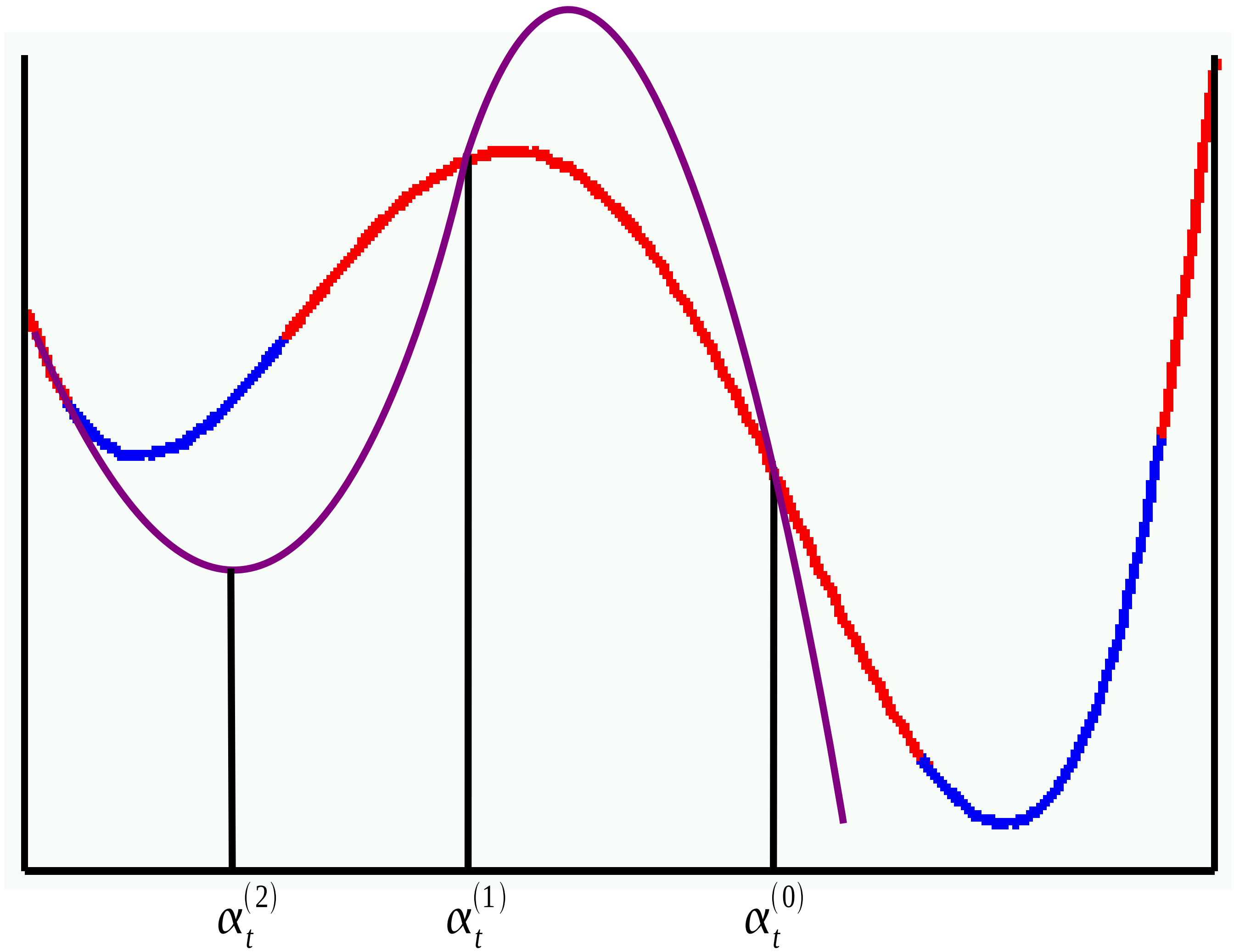
Step 1: Quadratic interpolation



Practical line search strategies

An alternative strategy ("Interpolating line search"):

Step 2 and following: Cubic interpolation



Part 5

Smooth unconstrained problems: Trust region algorithms

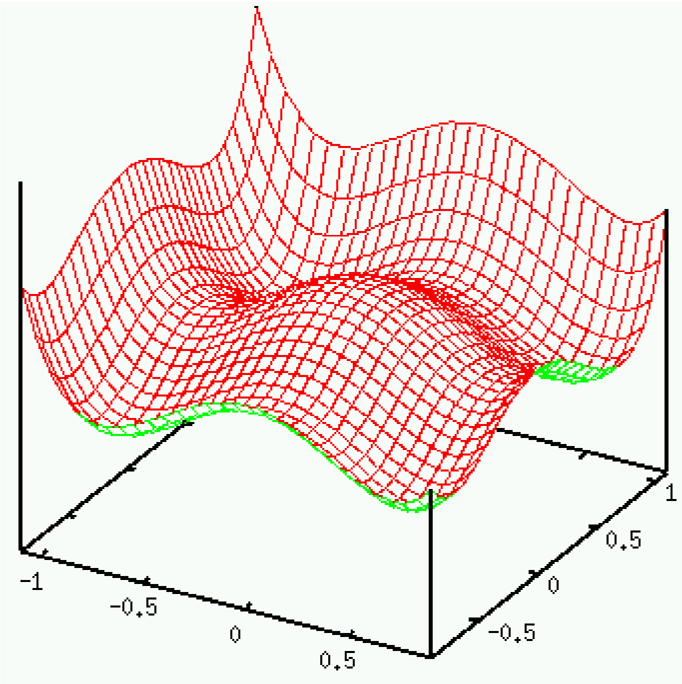
$$\text{minimize} \quad f(x)$$

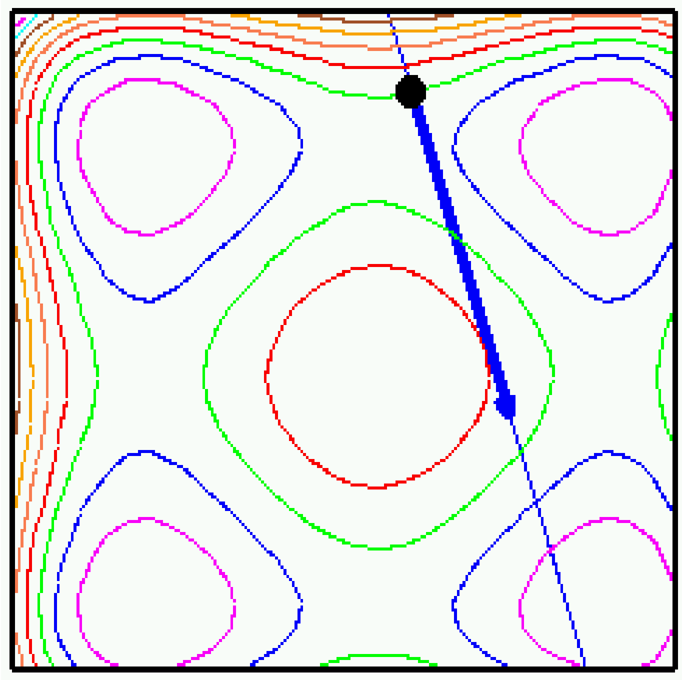
Line search vs. trust region algorithms

Line search algorithms:

Choose a relatively simple strategy to find a search direction

Put significant effort into finding an appropriate step length





Line search vs. trust region algorithms

Trust region algorithms:

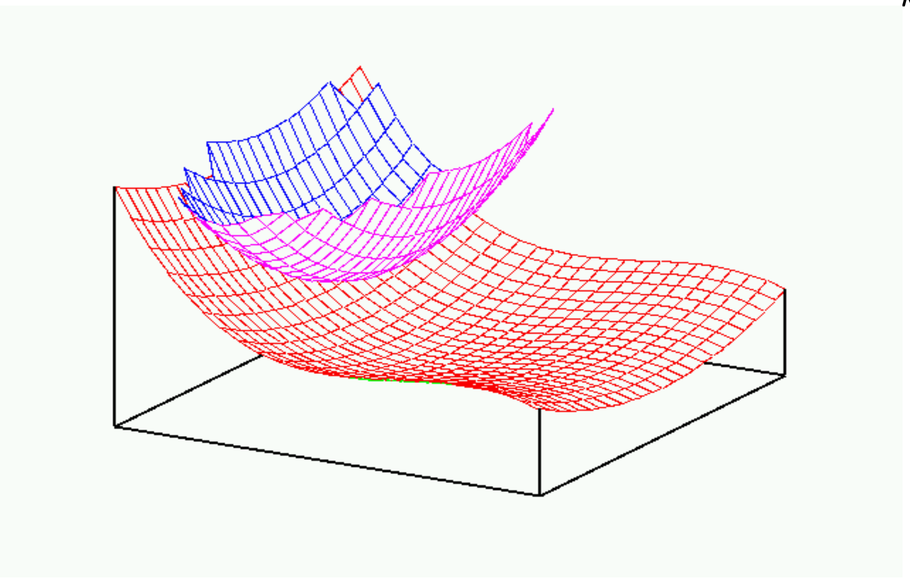
Choose a relatively simple strategy to determine a step length

Put significant effort into finding an appropriate search direction

Background:

In line search methods, we choose a direction based on a *local* approximation of the objective function

In other words: We try to predict the behavior of $f(x)$ far away from x_k by looking at f_k, g_k, H_k



That can not work if we are still far away from the solution! (Unless the function is almost quadratic everywhere.)

Trust region algorithms

Trust region algorithms:

Choose a relatively simple strategy to determine a step length

Put significant effort into finding an appropriate search direction

Alternative strategy:

Keep a number Δ_k around that indicates up to which distance we *trust* that our model $m_k(p)$ is a good approximation of $f(x_k+p_k)$

Find an update as follows:

$$p_k = \arg\min_p \; m_k(p) = f_k + g_k \cdot p + \frac{1}{2} p^T B p$$

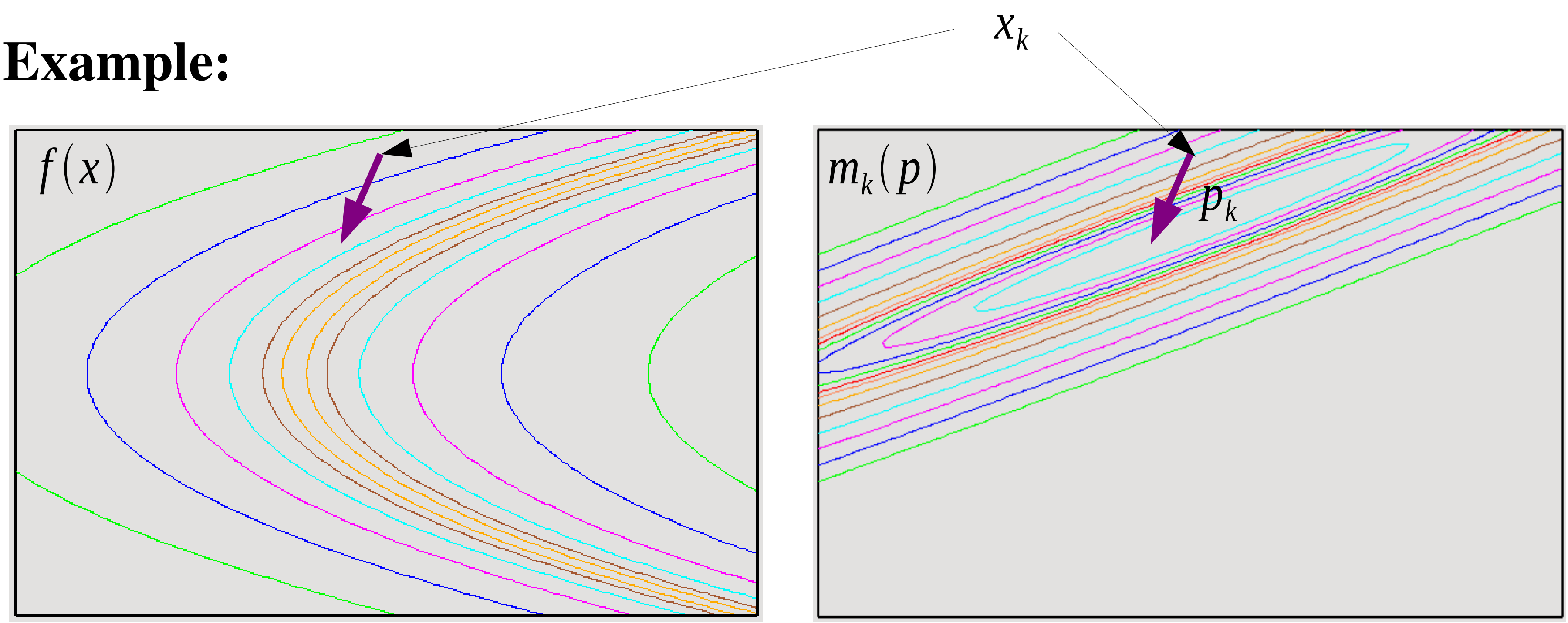
$$\text{such that} \quad \|p\| \leq \Delta_k$$

Accept this direction unconditionally, i.e. without line search:

$$x_{k+1} = x_k + p_k$$

Trust region algorithms

Example:



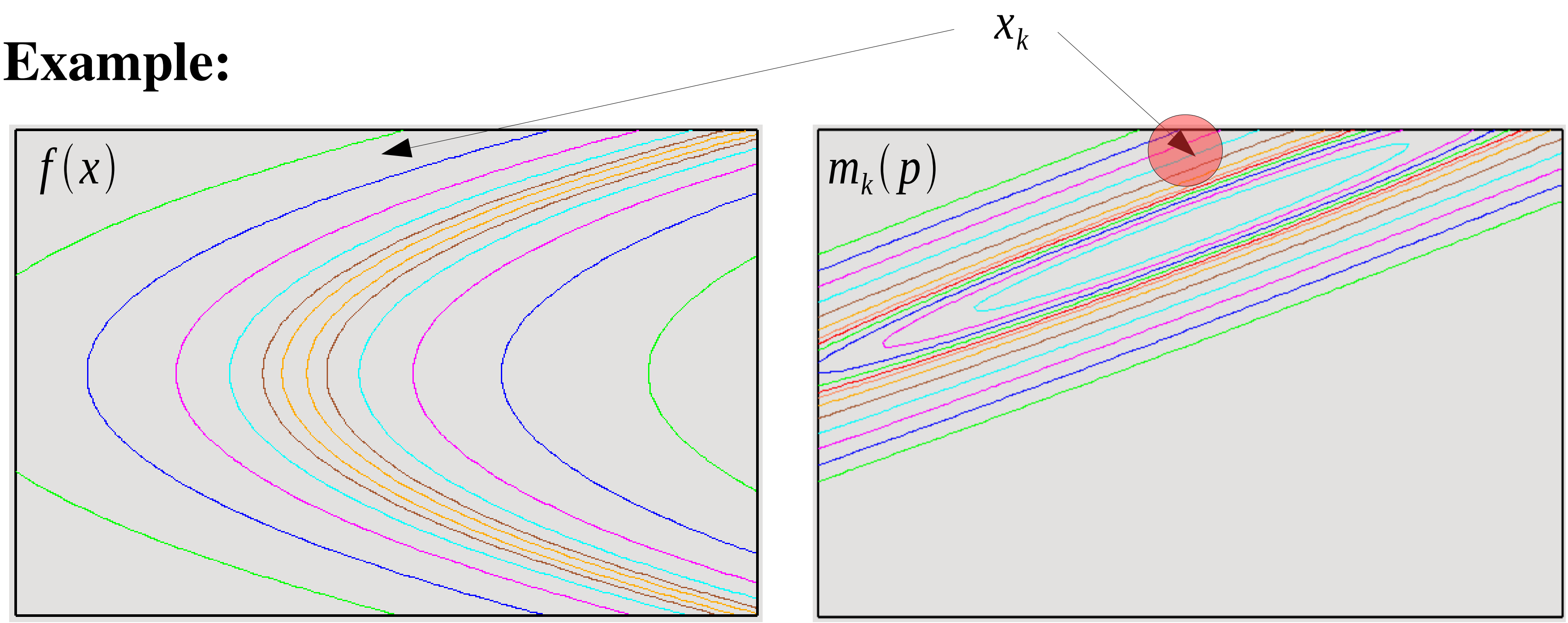
The line search Newton direction leads to the exact minimum of the approximating model function $m_k(p)$.

However, this is of little help because $m_k(p)$ is not approximate $f(x)$ well at these distances.

As a consequence, we need line search as a safe guard.

Trust region algorithms

Example:



Rather, decide how far we trust the model and stay within this radius!

Trust region algorithms

Basic trust region algorithm:

For $k=1,2,\ldots$:

•Compute update by finding an approximation $\tilde{p}_k$ to the solution of

$$\boxed{\begin{aligned} p_k &= \arg\min_p \; m_k(p) = f_k + g_k \cdot p + \frac{1}{2} p^T B_k p \\ &\quad \text{such that} \quad \|p\| \le \Delta_k \end{aligned}}$$

•Compute predicted improvement $PI = m_k(0) - m_k(\tilde{p}_k)$

•Compute actual improvement $AI = f(x_k) - f(x_k + \tilde{p}_k)$

•If $AI/PI < 1/4$ then $\Delta_{k+1} = \frac{1}{4}\|\tilde{p}_k\|$

$AI/PI > 3/4$ and $\|p_k\| = \Delta_k$ then $\Delta_{k+1} = 2\Delta_k$

•If $AI/PI > \eta$ for some $\eta \in [0,1/4)$ then $x_{k+1} = x_k + \tilde{p}_k$

else $x_{k+1} = x_k$

Trust region algorithms

Fundamental difficulty of trust region algorithms:

$$p_k = \arg\min_p \; m_k(p) = f_k + g_k \cdot p + \frac{1}{2} p^T B_k p$$

$$\text{such that} \quad \|p\| \leq \Delta_k$$

- This is not a trivial problem to solve!
- As with line search algorithms, we don't want to spend a lot of time finding the exact minimum of an approximate model.

- Trust region methods are all about finding cheap ways to approximate the solution of the problem above!

Trust region algorithms: The dogleg method

Find an approximation to the solution of:

$$p_k = \arg\min_p \ m_k(p) = f_k + g_k \cdot p + \frac{1}{2} p^T B_k p$$
$$\text{such that} \quad \|p\| \leq \Delta_k$$

Note:

If the trust region radius is small, then we get the "Cauchy point" in the steepest descent direction:

$$p_k \approx p_k^C = \tau \, p_k^{SD} \qquad \tau \in [0,1] \qquad p_k^{SD} = -\Delta_k \frac{g_k}{\|g_k\|}$$

p_k^C is the minimizer of *f(x)* in direction p_k^{SD}

If the trust region radius is large, then we get the quasi-Newton direction:

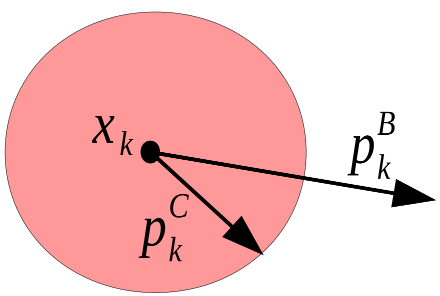
$$p_k = p_k^B = -B_k^{-1} g_k$$

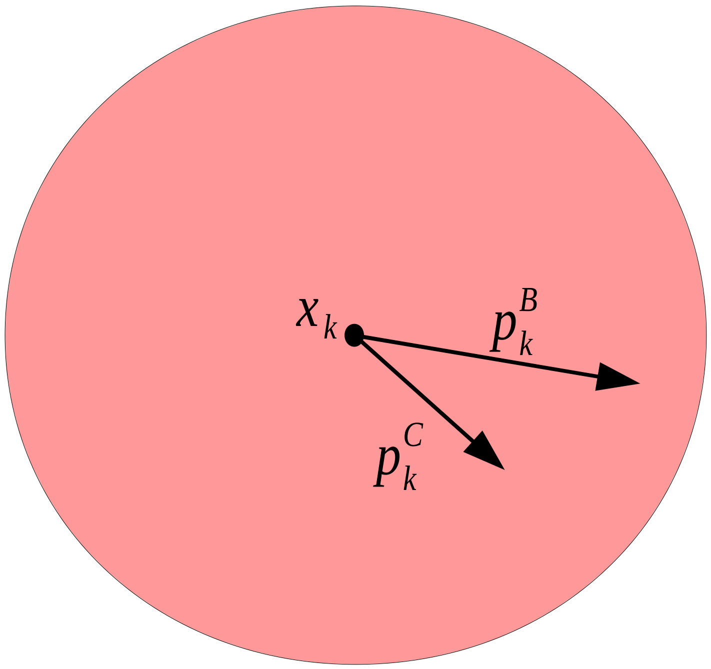
Trust region algorithms: The dogleg method

Find an approximation to the solution of:

$$p_k = \arg\min_p \; m_k(p) = f_k + g_k \cdot p + \frac{1}{2} p^T B_k p$$

$$\text{such that} \quad \|p\| \leq \Delta_k$$





$\Delta_k < \| p_k^B \|$

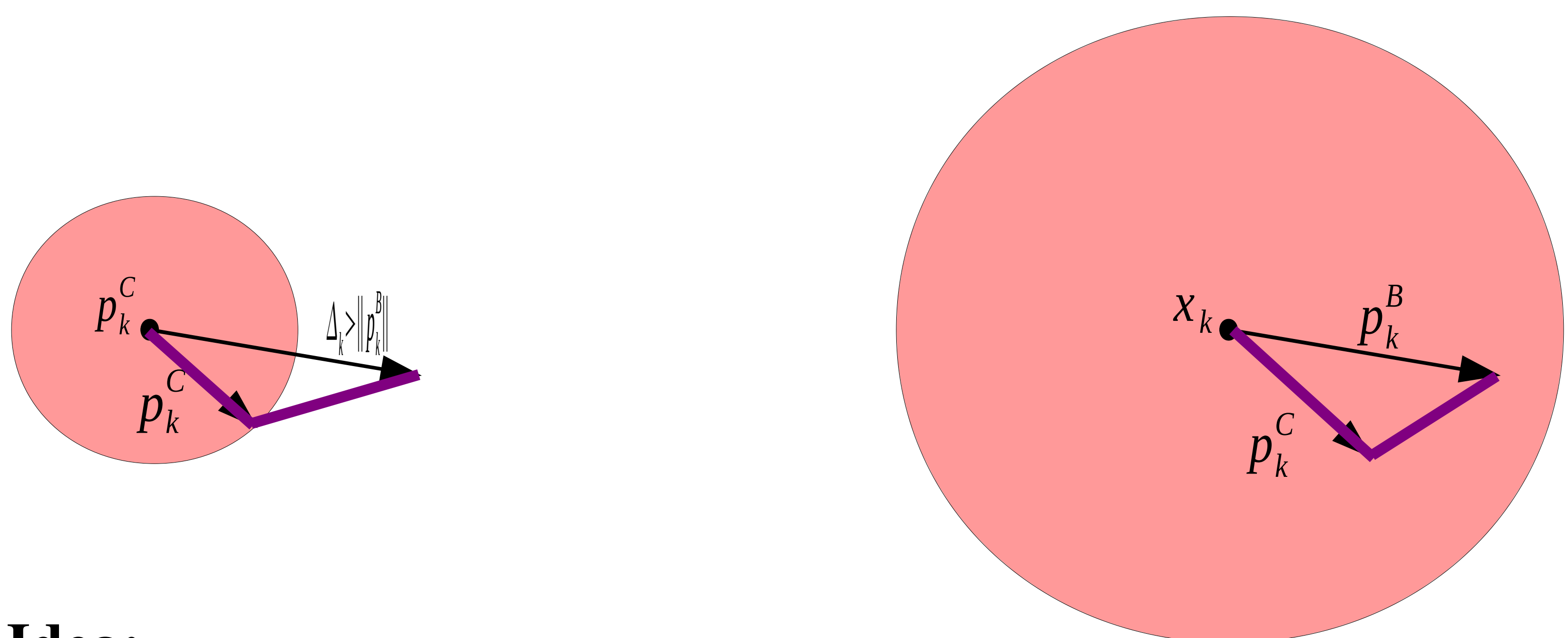
$\Delta_k > \| p_k^B \|$

Trust region algorithms: The dogleg method

Find an approximation to the solution of:

$$p_k = \arg\min_p \; m_k(p) = f_k + g_k \cdot p + \frac{1}{2} p^T B_k p$$

$$\text{such that} \quad \| p \| \leq \Delta_k$$



Idea:

Find the approximate solution $\tilde{p}_k$ along the "dogleg" line

$$x_k \rightarrow x_k + p_k^C \rightarrow x_k + p_k^B$$

Trust region algorithms: The dogleg method

Find an approximation to the solution of:

$$p_k = \arg\min_p \; m_k(p) = f_k + g_k \cdot p + \frac{1}{2} p^T B_k p$$

$$\text{such that} \quad \| p \| \le \Delta_k$$

In practice, the Cauchy point is difficult to compute because it requires a line search.

The dogleg method therefore uses not the minimizer p_k^C of *f(x)* along p_k^{SD} but the minimizer

$$p_k^U = -\frac{g_k^T g_k}{g_k^T B_k g_k} g_k$$

of

$$m_k(p) = f_k + g_k^T p + \frac{1}{2} p^T B_k p$$

The dogleg then runs along $x_k \;\rightarrow\; x_k + p_k^U \;\rightarrow\; x_k + p_k^B$

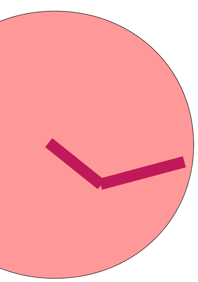
Trust region algorithms: The dogleg method

Find an approximation to the solution of:

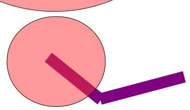
$$p_k = \arg\min_p \; m_k(p) = f_k + g_k \cdot p + \frac{1}{2} p^T B_k p$$

$$\text{such that} \quad \|p\| \leq \Delta_k$$

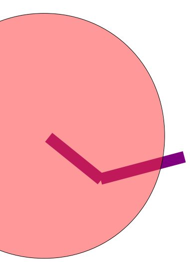
Dogleg algorithm:



If $p_k^B = -B_k^{-1} g_k$ satisfies $\|p_k^B\| < \Delta_k$ then set $\tilde{p}_k = p_k^B$



Otherwise, if $p_k^U = -\dfrac{g_k^T g_k}{g_k^T B_k g_k} g_k$ satisfies $\|p_k^U\| > \Delta_k$ then set $\tilde{p}_k = \dfrac{p_k^U}{\|p_k^U\|} \Delta_k$



Otherwise choose $\tilde{p}_k$ as the intersection point of the line $p_k^U \rightarrow p_k^B$ and the circle with radius Δ_k

Part 6

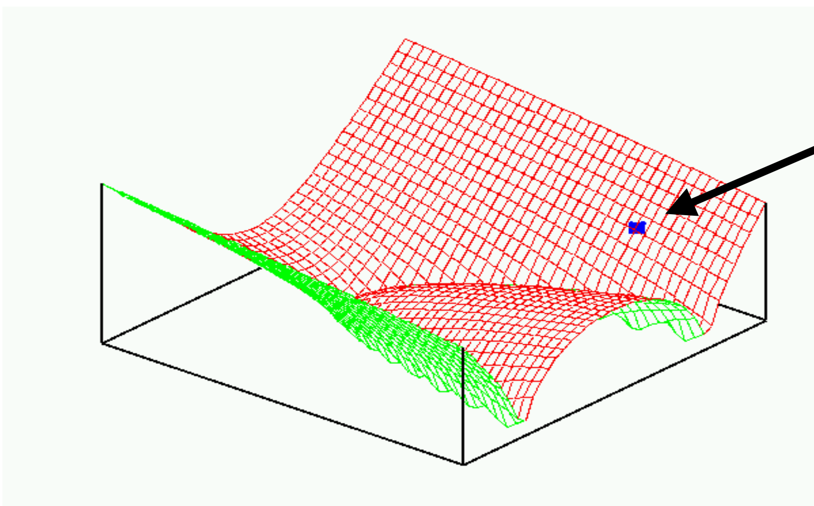
Practical aspects of Newton methods

$$\text{minimize} \quad f(x)$$

What if the Hessian is not positive definite

At the solution, the Hessian $\nabla^2 f(x^*)$ is positive definite. Since *f(x)* is smooth, the Hessian is positive definite if near the optimum.

However, this needs not be so far away from the optimum:



At initial point x_0 the Hessian is indefinite:

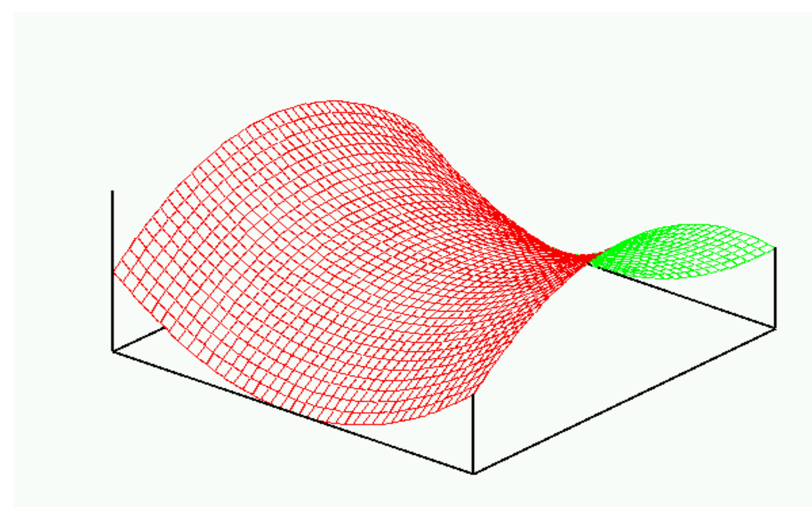
$$H_0 = \nabla^2 f(x_0) = \begin{pmatrix} -0.022 & 0.134 \\ 0.134 & -0.337 \end{pmatrix}$$

$$\lambda_1 = -0.386, \quad \lambda_2 = 0.027$$

Quadratic model

$$m_k(p) = f_k + g_k^T p + \frac{1}{2} p^T H_k p$$

has saddle point instead of minimum, Newton step is invalid!



What if the Hessian is not positive definite

Background: Search direction only useful if descent direction:

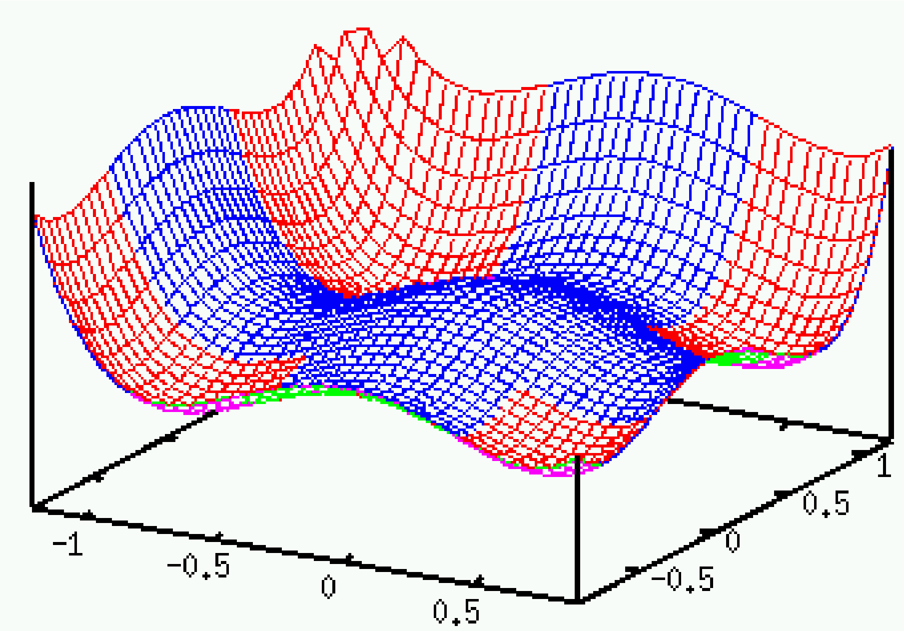
$$\nabla f(x_k)^T \cdot p_k < 0$$

Trivially satisfied for Gradient method, for Newton's method there holds:

$$p_k = -H_k^{-1} g_k \qquad \rightarrow \qquad g_k^T \cdot p_k = -g_k^T H_k^{-1} g_k < 0$$

Search direction only then a guaranteed descent direction, if *H* positive definite!

Otherwise search direction is direction to saddle point of quadratic model and might be a direction of *ascent*!



What if the Hessian is not positive definite

If the Hessian is not positive definite, then modify the quadratic model:

- retain as much information as possible;
- model should be convex, so that we can seek a minimum.

The general strategy then is to replace the quadratic model by a positive definite one:

$$m_k(p) = f_k + g_k^T p + \frac{1}{2} p^T \tilde{H}_k p$$

Here, $\tilde{H}_k$ is a suitable modification of the exact Hessian $H_k = \nabla^2 f(x_k)$ so that $\tilde{H}_k$ is positive definite.

Note: To retain ultimate quadratic convergence, we need that

$$\tilde{H}_k \to H_k \qquad \text{as} \qquad x_k \to x^*$$

What if the Hessian is not positive definite

The **Levenberg-Marquardt** modification:

Choose

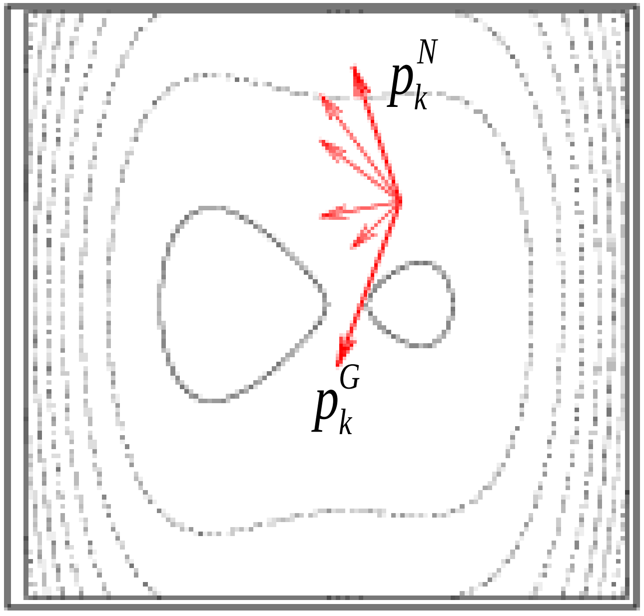
$$\tilde{H}_k = H_k + \tau I \qquad \tau > -\lambda_i$$

so that the minimum of

$$m_k(p) = f_k + g_k^T p + \frac{1}{2} p^T \tilde{H}_k p$$

lies at

$$p_k = -\tilde{H}_k^{-1} g_k = -(H_k + \tau I)^{-1} g_k$$



Note: Search direction is mixture between Newton direction and gradient.

Note: Close to the solution the Hessian must become positive definite and we can choose $\tau = 0$

What if the Hessian is not positive definite

The eigenvalue modification strategy:

Since H is symmetric, it has a complete set of eigenvectors:

$$H_k = \nabla^2 f(x_k) = \sum_i \lambda_i v_i v_i^T$$

Therefore replace the quadratic model by a positive definite one:

$$m_k(p) = f_k + g_k^T p + \frac{1}{2} p^T \tilde{H}_k p$$

with

$$\tilde{H}_k = \sum_i \max\{\lambda_i, \epsilon\} \; v_i v_i^T$$

Note: Only modify the Hessian in directions of negative curvature.

Note: Close to the solution, all eigenvalues become positive and we get again the original Newton matrix.

What if the Hessian is not positive definite

One problem with the modification

$$\tilde{H}_k = \sum_i \max\{\lambda_i, \epsilon\} \; v_i v_i^T$$

is that the search direction is given by

$$p_k = -\tilde{H}_k^{-1} g_k = \sum_i \frac{1}{\max\{\lambda_i, \epsilon\}} \; v_i \left(v_i^T g_k\right)$$

that is search direction has *large* component (of size $1/\varepsilon$) in direction of modified curvatures!

An alternative that avoids this is to use

$$\tilde{H}_k = \sum_i |\lambda_i| v_i v_i^T$$

What if the Hessian is not positive definite

Theorem: Using full step length and either of the Hessian modifications

$$\tilde{H}_k = H_k + \tau I \qquad \tau > -\lambda_i$$

$$\tilde{H}_k = \sum_i \max\{\lambda_i, \epsilon\} \; v_i v_i^T$$

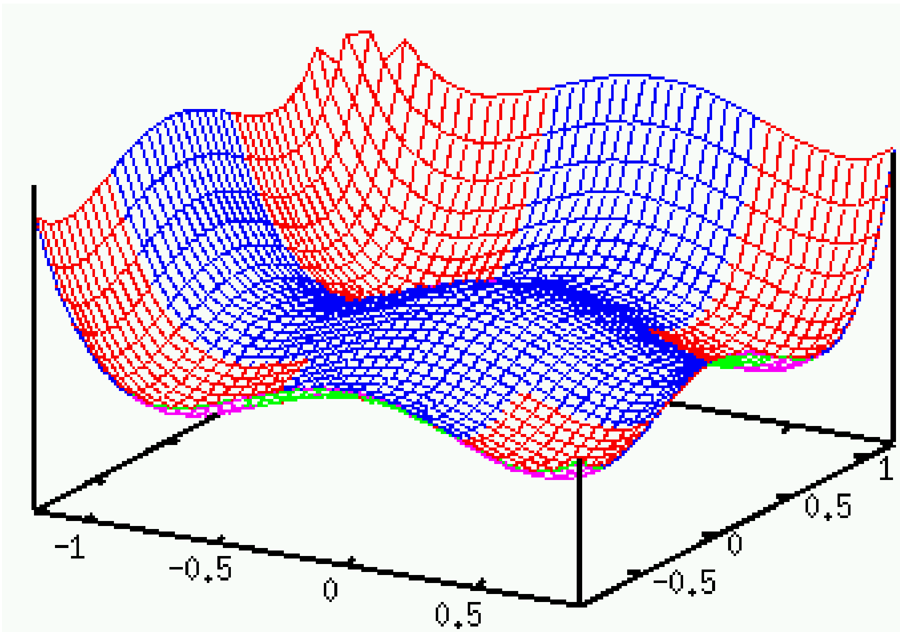
we have that if $x_k \to x^*$ and if $f \in C^{2,1}$ then convergence happens with quadratic rate.

Proof: Since f is twice continuously differentiable, there is a k such that x_k is close enough to x^* that H_k is positive definite. When that is the case, then

$$\tilde{H}_k = H_k$$

for all following iterations, providing the quadratic convergence rate of the full step Newton method.

What if the Hessian is not positive definite



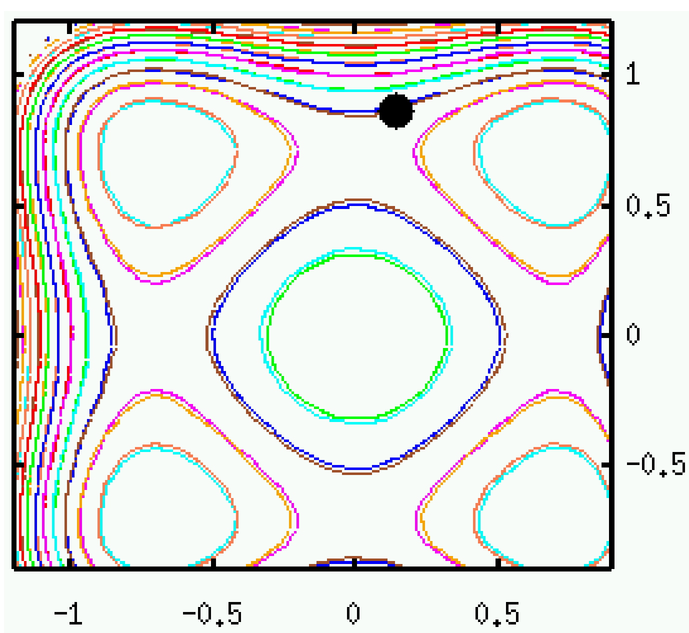
Example:

$$f(x,y) = x^4 - x^2 + y^4 - y^2$$

Blue regions indicate that Hessian

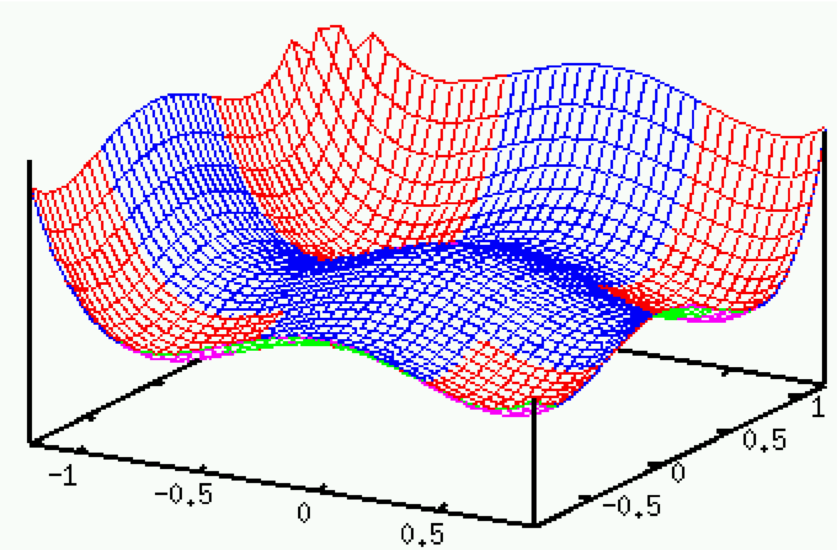
$$\nabla^2 f(x,y) = \begin{pmatrix} 12x^2-2 & 0 \\ 0 & 12y^2-2 \end{pmatrix}$$

is not positive definite.



minima at $x = \frac{\pm\sqrt{2}}{2,}\ y = \frac{\pm\sqrt{(2)}}{2}$

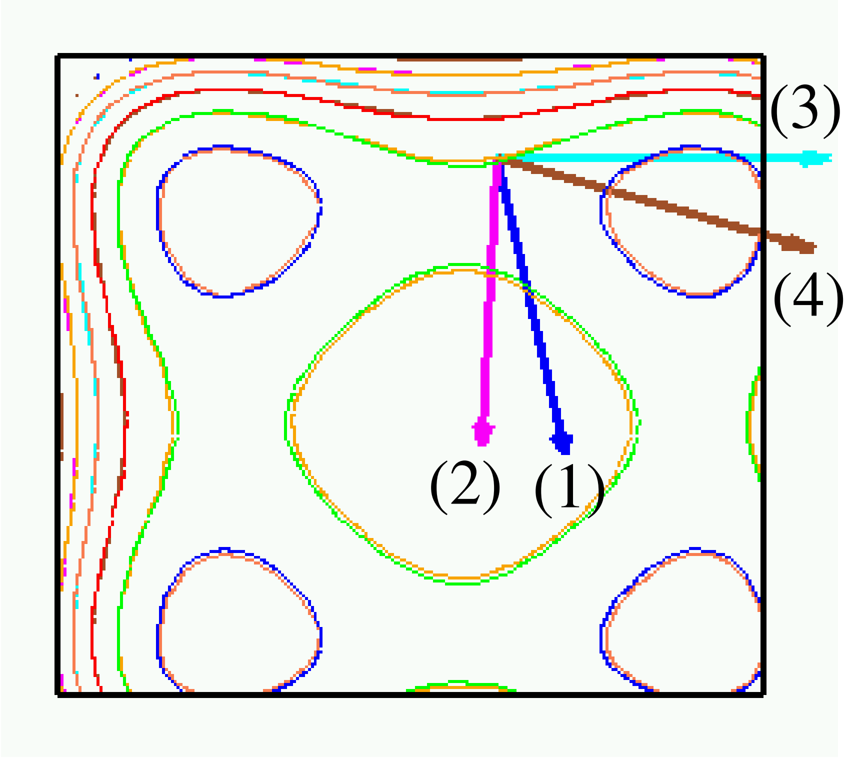
What if the Hessian is not positive definite



Starting point:

$$x_0 = 0.1 \qquad y_0 = 0.87$$

$$H_0 = \begin{pmatrix} -1.88 & 0 \\ 0 & 7.08 \end{pmatrix}$$



Negative gradient

Unmodified Hessian search direction

Search direction with eigenvalue modified Hessian ($\varepsilon = 10^{-6}$)

Search direction with shifted Hessian ($\tau = 2.5$; search direction only good by lucky choice of τ)

Truncated Newton methods

In any Newton or Trust Region method, we have to solve an equation of the sort

$$H_k p_k = -g_k$$

or potentially with a modified Hessian:

$$\tilde{H}_k p_k = -g_k$$

Oftentimes, computing the Hessian is more expensive than inverting it, but not always.

Question: Could we possibly get away with only approximately solving this problem, i.e. finding

$$p_k \approx -H_k^{-1} g_k$$

with suitable conditions on how accurate the approximation is?

Truncated Newton methods

Example: Since the Hessian (or a modified version) is a positive definite matrix, we may want to solve

$$H_k\, p_k = -g_k$$

using an iterative method such as the Conjugate Gradient method, Gauss-Seidel, Richardson iteration, SSOR, etc etc.

While all these methods eventually converge to the exact Newton direction, we may want to *truncate* this iteration at one point.

Question: When can we terminate this iteration?

Truncated Newton methods

Theorem 1: Let $\hat{p}_k$ be an approximation to the Newton direction defined by

$$H_k p_k = -g_k$$

and let there be a sequence of numbers $\{\eta_k\}, \eta_k < 1$ so that

$$\frac{\|g_k + H_k \hat{p}_k\|}{\|g_k\|} \leq \eta_k < 1$$

Then if $x_k \to x^*$ then the full step Newton method converges with linear order.

Truncated Newton methods

Theorem 2: Let $\hat{p}_k$ be an approximation to the Newton direction defined by

$$H_k p_k = -g_k$$

and let there be a sequence of numbers $\{\eta_k\}, \eta_k < 1, \eta_k \to 0$ so that

$$\frac{\|g_k + H_k \hat{p}_k\|}{\|g_k\|} \le \eta_k < 1$$

Then if $x_k \to x^*$ then the full step Newton method converges with superlinear order.

Truncated Newton methods

Theorem 3: Let $\hat{p}_k$ be an approximation to the Newton direction defined by

$$H_k p_k = -g_k$$

and let there be a sequence of numbers $\{\eta_k\}, \eta_k < 1, \eta_k = O(\|g_k\|)$ so that

$$\frac{\|g_k + H_k \hat{p}_k\|}{\|g_k\|} \leq \eta_k < 1$$

Then if $x_k \to x^*$ then the full step Newton method converges with quadratic order.

Part 7

Quasi-Newton update formulas

$$B_{k+1} = B_k + \ldots$$

Quasi-Newton update formulas

Observation 1:

Computing the exact Hessian to determine the Newton search direction

$$H_k\, p_k = -g_k$$

is expensive, and sometimes impossible.

It *at least* doubles the effort per iteration because we need not only the first but also the second derivative of $f(x)$.

It also requires us to solve a linear system for the search direction.

Quasi-Newton update formulas

Observation 2:

We know that we can get superlinear convergence if we choose the update p_k using

$$B_k p_k = -g_k$$

instead of

$$H_k p_k = -g_k$$

under certain conditions on the matrix B_k.

Quasi-Newton update formulas

Question:

- Maybe it is possible to find matrices B_k for which:
  - Computing B_k is cheap and requires no additional function evaluations
  - Solving
    $$B_k p_k = -g_k$$
    for p_k is cheap
  - The resulting iteration still converges with superlinear order.

Motivation of ideas

Consider a function *p(x)*.

The **Fundamental Theorem of Calculus** tells us that

$$p(z) - p(x) = \nabla p(\xi)^T (z - x)$$

for some $\xi = x + t(z - x),\ t \in [0,1]$

Let's apply this to $p(x) = \nabla f(x),\quad z = x_k,\ x = x_{k-1}$:

$$\begin{aligned}\nabla f(x_k) - \nabla f(x_{k-1}) = g_k - g_{k-1} &= \nabla^2 f(x_k - t\alpha p_k)(x_k - x_{k-1}) \\ &= \tilde{H}(x_k - x_{k-1})\end{aligned}$$

Let us denote $y_{k-1} = g_k - g_{k-1}, s_{k-1} = x_k - x_{k-1}$ then this reads

$$\tilde{H} s_{k-1} = y_{k-1}$$

with an "average" Hessian $\tilde{H}$.

Motivation of ideas

Requirements:

- We seek a matrix B_{k+1} so that
- The "secant condition" holds:

$$B_{k+1} s_k = y_k$$

- B_{k+1} is symmetric
- B_{k+1} is positive definite
- B_{k+1} changes minimally from B_k
- The update equation is easy to solve for

$$p_{k+1} = -B_{k+1}^{-1} g_{k+1}$$

Davidon-Fletcher-Powell

The DFP update formula:

Given B_k define B_{k+1} by

$$B_{k+1} = (I - \gamma\, y_k s_k^T) B_k (I - \gamma\, s_k y_k^T) + \gamma\, y_k y_k^T$$

$$\gamma_k = \frac{1}{y_k^T s_k}$$

This satisfies the conditions:

- It is symmetric and positive definite
- It is among all possible matrices the one that minimizes

$$\| \tilde{H}^{-1/2} (B_{k+1} - B_k) \tilde{H}^{-1/2} \|_F$$

- It satisfies the secant condition $B_{k+1} s_k = y_k$

Broyden-Fletcher-Goldfarb-Shanno

The BFGS update formula:

Given B_k define B_{k+1} by

$$B_{k+1} = B_k - \frac{B_k s_k s_K^T B_k}{s_k^T B_k s_K} + \frac{y_k y_k^T}{y_k^T s_k}$$

This satisfies the conditions:

- It is symmetric and positive definite
- It is among all possible matrices the one that minimizes

$$\|\tilde{H}^{1/2}(B_{k+1}^{-1} - B_k^{-1})\tilde{H}^{1/2}\|_F$$

- It satisfies the secant condition $B_{k+1} s_k = y_k$

Broyden-Fletcher-Goldfarb-Shanno

So far:

- We seek a matrix B_{k+1} so that
- The secant condition holds:

$$B_{k+1} s_k = y_k$$

- B_{k+1} is symmetric
- B_{k+1} is positive definite
- B_{k+1} changes minimally from B_k in some sense
- The update equation is easy to solve for

$$p_k = -B_k^{-1} g_k$$

DFP and BFGS

Now a miracle happens:

For the DFP formula:

$$B_{k+1} = (I - \gamma_k y_k s_k^T) B_k (I - \gamma_k s_k y_k^T) + \gamma_k y_k y_k^T, \qquad \gamma_k = \frac{1}{y_k^T s_k}$$

$$B_{k+1}^{-1} = B_k^{-1} - \frac{B_k^{-1} y_k y_k^T B_k^{-1}}{y_k^T B_k^{-1} y_k} + \frac{s_k s_k^T}{y_k^T s_k}$$

For the BFGS formula:

$$B_{k+1} = B_k - \frac{B_k s_k s_K^T B_k}{s_k^T B_k s_k} + \frac{y_k y_k^T}{y_k^T s_k}$$

$$B_{k+1}^{-1} = (I - \rho_k s_k y_k^T) B_k^{-1} (I - \rho_k y_k s_k^T) + \rho_k s_k s_k^T, \qquad \rho_k = \frac{1}{y_k^T s_k}$$

This makes computing the next update very cheap!

DFP + BFGS = Broyden class

What if we mixed:

$$B_{k+1}^{DFP} = (I - \gamma_k y_k s_k^T) B_k (I - \gamma_k s_k y_k^T) + \gamma_k y_k y_k^T, \qquad \gamma_k = \frac{1}{y_k^T s_k}$$

$$B_{k+1}^{BFGS} = B_k - \frac{B_k s_k s_K^T B_k}{s_k^T B_k s_k} + \frac{y_k y_k^T}{y_k^T s_k}$$

$$B_{k+1} = \phi_k B_{k+1}^{DFP} + (1 - \phi_k) B_k^{BFGS}$$

This is called the "Broyden class" of update formulas.

The class of Broyden methods with $0 \le \phi_k \le 1$ is called the "restricted Broyden class".

DFP + BFGS = Broyden class

Theorem: Let $f \in C^2$, let x_0 be a starting point so that the set

$$\Omega = \{x : f(x) \leq f(x_0)\}$$

is convex. Let B_0 be any symmetric positive definite matrix. Then

$$x_k \to x^*$$

for any sequence x_k generated by a quasi-Newton method that uses a Hessian update formula by any member of the restricted Broyden class with the exception of the DFP method $(\phi_k = 1)$.

DFP + BFGS = Broyden class

Theorem: Let $f \in C^{2,1}$. Assume the BFGS updates converge, then

$$x_k \to x^*$$

with superlinear order.

Practical BFGS: Starting matrix

Question: How do we choose the initial matrix B_0 or B_0^{-1} ?

Observation 1: The theorem stated that we will eventually converge for any symmetric, positive definite starting matrix.

In particular, we could choose a multiple of the identity matrix

$$B_0 = \beta I, \quad B_0^{-1} = \frac{1}{\beta} I$$

Observation 2: If β is too small, then

$$p_0 = -B_0^{-1} g_0 = -\frac{1}{\beta} g_0$$

is too large, and we need many trials in line search to find a suitable step length.

Observation 3: The matrices B should approximate the Hessian matrix, so they at least need to have the same physical units.

Practical BFGS: Starting matrix

Practical approaches:

Strategy 1: Compute the first gradient g_0, choose a "typical" step length δ , then set

$$B_0 = \frac{\|g_0\|}{\delta} I , \quad B_0^{-1} = \frac{\delta}{\|g_0\|} I$$

so that we get

$$p_0 = -B_0^{-1} g_0 = -\delta \frac{g_0}{\|g_0\|}$$

Strategy 2: Approximate the true Hessian somehow. For example, do one step with the heuristic above, choose

$$B_0 = \frac{y_1^T y_1}{y_1^T s_1} I , \quad B_0^{-1} = \frac{y_1^T s_1}{y_1^T y_1} I$$

and start over again.

Practical BFGS: Limited Memory BFGS (LM-BFGS)

Observation: The matrices

$$B_{k+1} = B_k - \frac{B_k s_k s_K^T B_k}{s_k^T B_k s_k} + \frac{y_k y_k^T}{y_k^T s_k}$$

$$B_{k+1}^{-1} = (I - \rho_k s_k y_k^T) B_k^{-1} (I - \rho_k y_k s_k^T) + \rho_k s_k s_k^T, \qquad \rho_k = \frac{1}{y_k^T s_k}$$

are full, even if the true Hessian is sparse.

Consequence:

We need to compute all n^2 entries, and store them.

Practical BFGS: Limited Memory BFGS (LM-BFGS)

Solution: Note that in the *k*th iteration, we can write

$$B_k^{-1} = V_{k-1}^T B_{k-1}^{-1} V_{k-1} + \rho_{k-1} s_{k-1} s_{k-1}^T$$

$$\text{with } \rho_{k-1} = \frac{1}{y_{k-1}^T s_{k-1}}, V_{k-1} = (I - \rho_{k-1} y_{k-1} s_{k-1}^T)$$

We can expand this recursively:

$$\begin{aligned} B_k^{-1} &= V_{k-1}^T B_{k-1}^{-1} V_{k-1} + \rho_{k-1} s_{k-1} s_{k-1}^T \\ &= V_{k-1}^T V_{k-2}^T B_{k-2}^{-1} V_{k-2} V_{k-1} \\ &\quad + \rho_{k-2} V_{k-1}^T s_{k-1} s_{k-2}^T V_{k-1} + \rho_{k-1} s_{k-1} s_{k-1}^T \\ &= \ldots \\ &= \left[V_{k-1}^T \cdots V_1^T \right] B_0^{-1} \left[V_1 \cdots V_{k-1} \right] \\ &\quad + \sum_{j=1}^{k} \rho_{k-j} \left\{ \left[V_{k-1}^T \cdots V_{k-j+1}^T \right] s_{k-j} s_{k-j}^T \left[V_{k-j+1} \cdots V_{k-1} \right] \right\} \end{aligned}$$

Consequence: We need only store *kn* entries.

Practical BFGS: Limited Memory BFGS (LM-BFGS)

Problem: *kn* elements may still be quite a lot if we need many iterations. Forming the product with this matrix will then also be expensive.

Solution: Limit memory and CPU time by only storing the last *m* updates:

$$B_k^{-1} = \left[V_{k-1}^T \cdots V_{k-m}^T\right] B_{0,k}^{-1} \left[V_{k-m} \cdots V_{k-1}\right] + \sum_{j=1}^{m} \rho_{k-j} \left\{ \left[V_{k-1}^T \cdots V_{k-j+1}^T\right] s_{k-j} s_{k-j}^T \left[V_{k-j+1} \cdots V_{k-1}\right] \right\}$$

Consequence: We need only store *mn* entries and multiplication with this matrix requires $2mn+O(m^3)$ operations.

Practical BFGS: Limited Memory BFGS (LM-BFGS)

$$B_k^{-1} = \left[V_{k-1}^T \cdots V_{k-m}^T\right] B_{0,k}^{-1} \left[V_{k-m} \cdots V_{k-1}\right]$$
$$+ \sum_{j=1}^{m} \rho_{k-j} \left\{ \left[V_{k-1}^T \cdots V_{k-j+1}^T\right] s_{k-j} s_{k-j}^T \left[V_{k-j+1} \cdots V_{k-1}\right] \right\}$$

In practice:

- Initial matrix can be chosen independently in each iteration; typical approach is again

$$B_{0,k}^{-1} = \frac{y_{k-1}^T s_{k-1}}{y_{k-1}^T y_{k-1}} I$$

- Typical values for *m* are between 3 and 30.

Parts 1-7

Summary of methods for smooth unconstrained problems

$$\text{minimize} \quad f(x)$$

Summary

Newton's method is unbeatable in terms of speed of convergence

However: to converge, one needs
- a line search method, using conditions like the Wolfe conditions
- to modify the Hessian matrix whenever it is not positive definite

Newton's method can be expensive or infeasible if
- computing second derivatives is complicated
- the number of variables is large

Quasi-Newton methods such as LM-BFGS can help in this case:
- need only the computation of first derivatives
- need little memory and no explicit matrix inversions
- but typically converge slower than Newton, at best superlinearly

Trust region methods are an alternative to Newton's method but share the same drawbacks

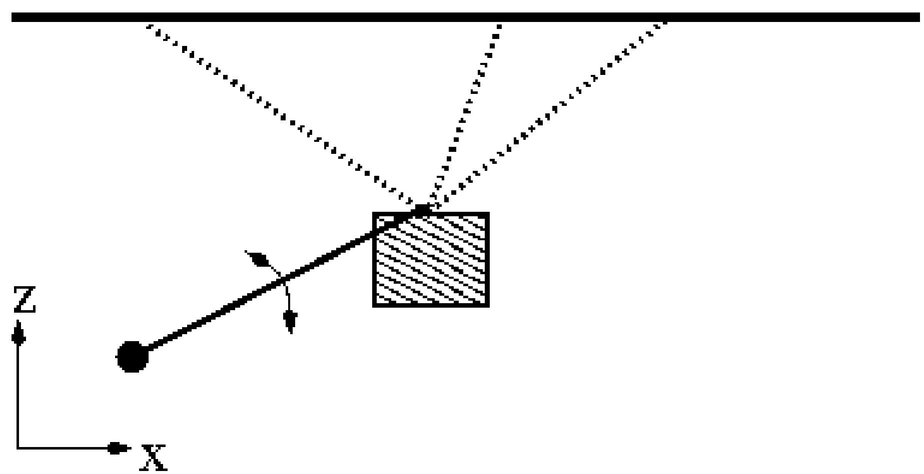
Part 8

Equality-constrained Problems

$$\begin{aligned} \text{minimize} \quad & f(x) \\ & g_i(x) = 0, \qquad i = 1, \dots, n_e \end{aligned}$$

An example

Consider the example of the body suspended from a ceiling with springs, but this time with an additional rod of fixed length attached to a fixed point:



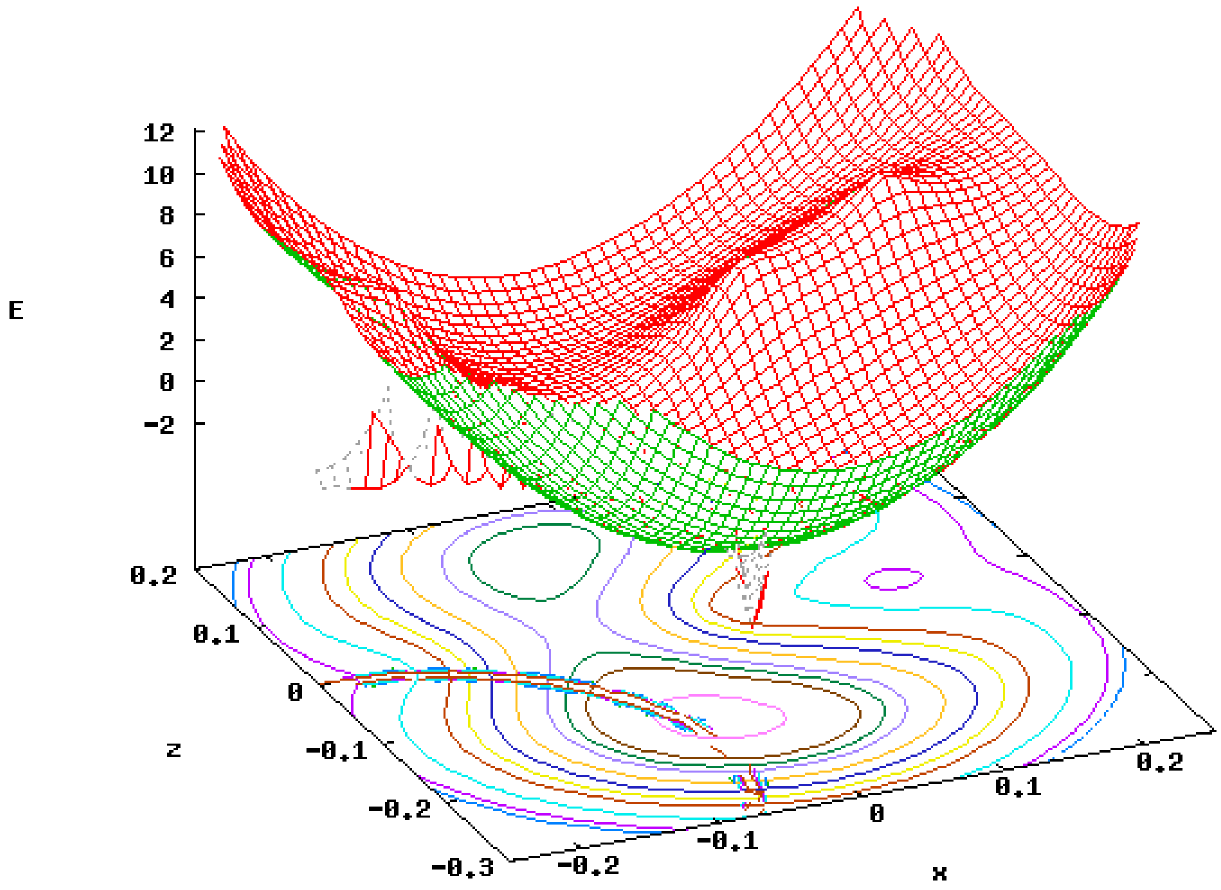
To find the position of the body we now need to solve the following problem:

$$\text{minimize} \quad f(\vec{x}) = E(x,z) = \sum_i E_{\text{spring},i}(x,z) + E_{\text{pot}}(x,z)$$

$$\|\vec{x} - \vec{x}_0\| - L_{\text{rod}} \;\; = \;\; 0$$

An example

We can gain some insight into the problem by plotting the energy as a function of *(x,z)* along with the constraint:



Definitions

We call this the standard form of equality constrained problems:

$$\text{minimize}_{x \in D \subset R^n} \quad f(x)$$
$$g_i(x) = 0, \quad i = 1 \ldots n_e$$

We will also frequently write this as follows, implying equality elementwise:

$$\text{minimize}_{x \in D \subset R^n} \quad f(x)$$
$$g(x) = 0$$

Definitions

A trivial reformulation of the problem is obtained by defining the *feasible set*:

$$\Omega = \{ x \in R^n : \; g(x) = 0 \}$$

Then the original problem is equivalently recast as

$$\text{minimize}_{x \in D \cap \Omega \subset R^n} \; f(x)$$

Note 1: This reformulation is not of much practical interest.

Note 2: The feasible set can be continuous or discrete. It can also be empty if the constraints are mutually incompatible. In the following we will always assume that it is continuous and non-empty.

The quadratic penalty method

Observation: The solution of

$$\text{minimize}_{x \in D \subset R^n} \quad f(x)$$
$$g(x) = 0$$

must lie within the feasible set where *g(x)=0*.

Idea: Let's *relax* the constraint and allow to search also in the vicinity where *g(x)* is small but not zero. However, make sure that the objective function becomes very large if far away from the feasible set:

$$\text{minimize}_{x \in D \subset R^n} \quad Q_\mu(x) = f(x) + \frac{1}{2\mu}\|g(x)\|^2$$

$Q_\mu(x)$ is called the *quadratic relaxation* of the constrained minimization problem. μ is the *penalty parameter.*

The quadratic penalty method

Why is *$Q_\mu(x)$* called *relaxation* of the constrained minimization problem with *$f(x)$, $g(x)$*?

Consider the original problem

$$\text{minimize} \quad f(\vec{x}) = E(x,z) = \sum_i E_{\text{spring},i}(x,z) + E_{\text{pot}}(x,z)$$

$$\|\vec{x}-\vec{x}_0\| - L_{\text{rod}} = 0$$

with relaxation

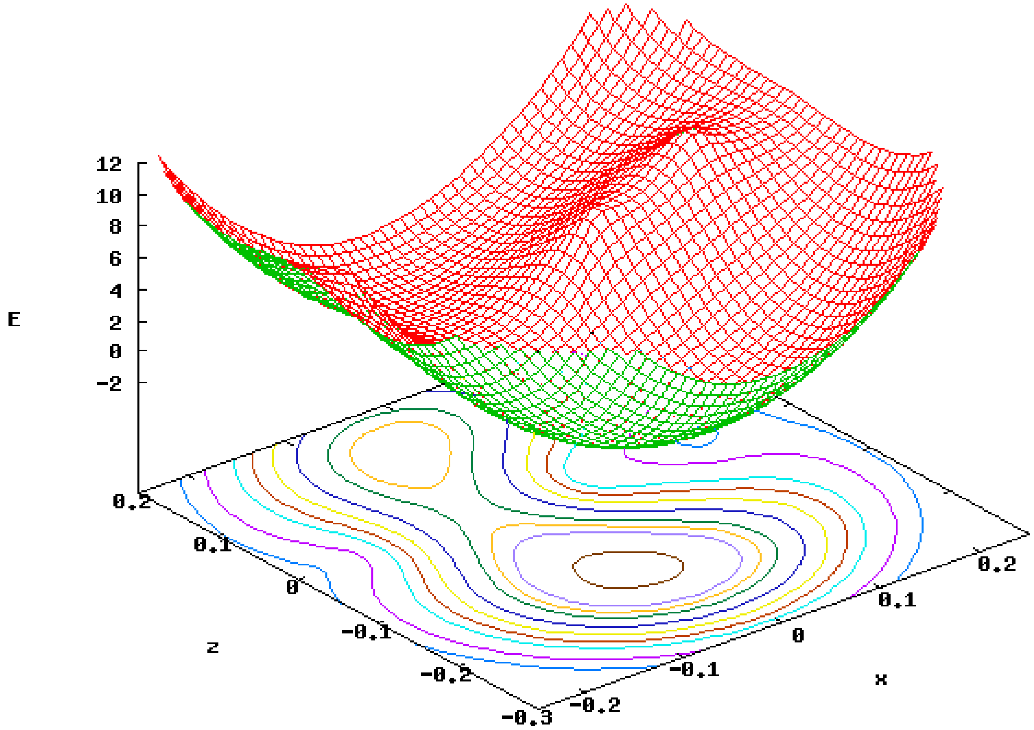
$$Q_\mu(\vec{x}) = E(x,z) + \frac{1}{2\mu}\left(\|\vec{x}-\vec{x}_0\| - L_{\text{rod}}\right)^2$$

If instead of a fixed rod we had a spring in this place with constant $\tilde{D}$ then we would have an unconstrained problem with objective function

$$\tilde{f}(\vec{x}) = E(x,z) + \frac{1}{2}\tilde{D}\left(\|\vec{x}-\vec{x}_0\| - L_{\text{rod}}\right)^2$$

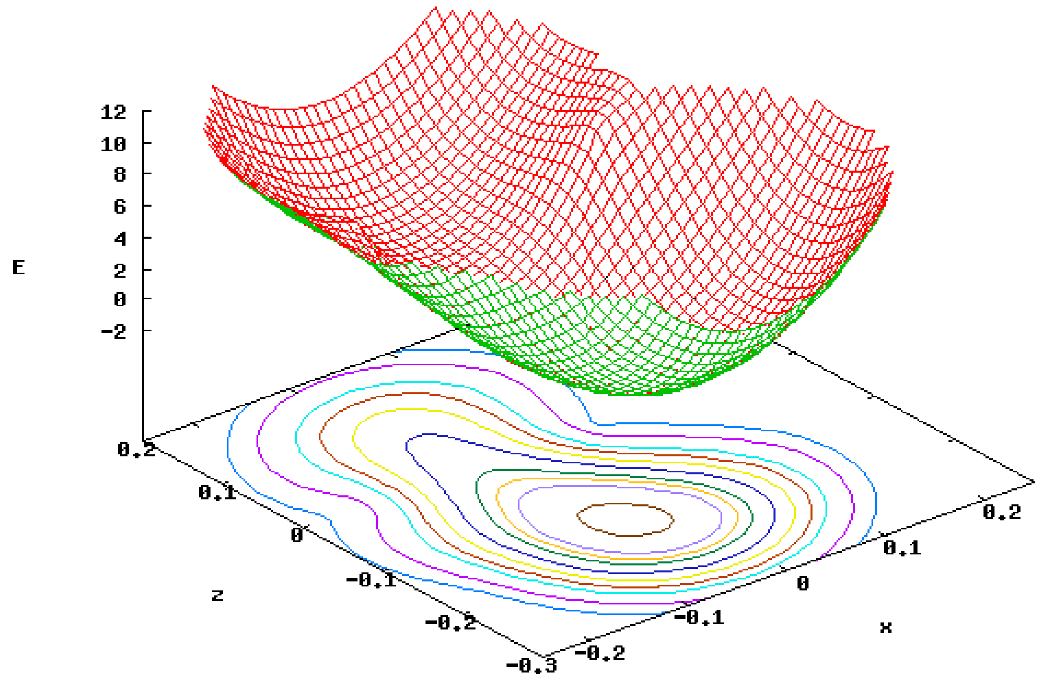
The quadratic penalty method

Example: $Q_\mu(x)$ with μ=infinity



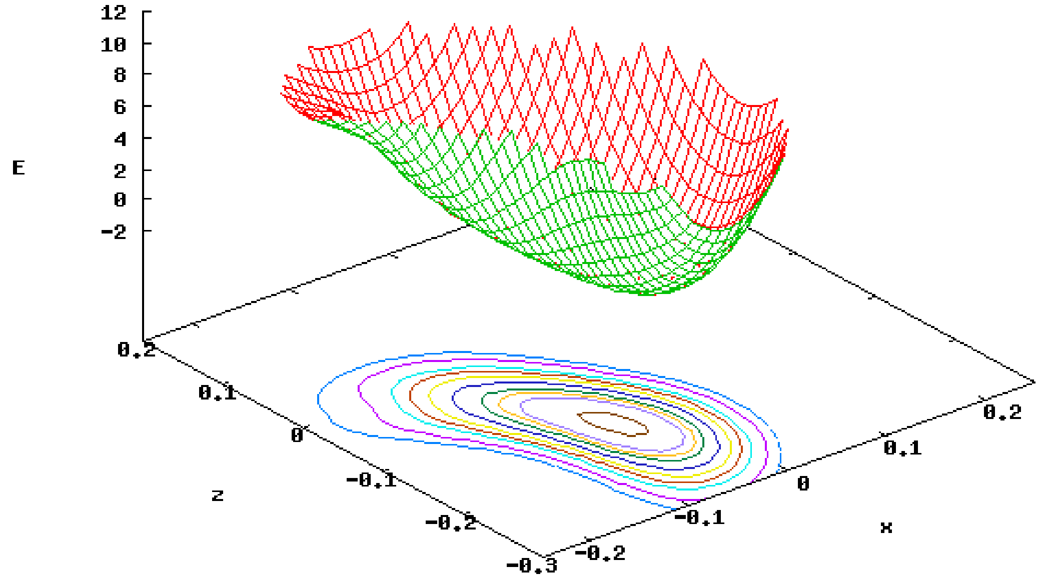
The quadratic penalty method

Example: $Q_\mu(x)$ with μ=*0.01*



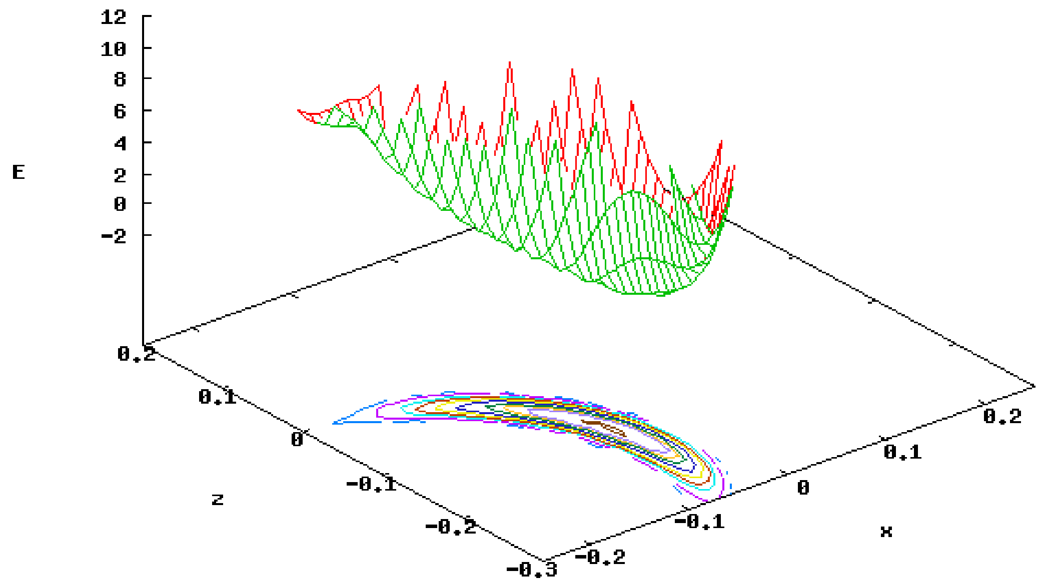
The quadratic penalty method

Example: $Q_\mu(x)$ with $\mu=0.001$



The quadratic penalty method

Example: $Q_\mu(x)$ with $\mu=0.00001$



The quadratic penalty method

Algorithm:

Given x_0^{start}, $\{\mu_t\} \to 0$, $\{\tau_t\} \to 0$

For t=0,1, 2, ...:

Find an approximation $\tilde{x}_t^*$ to the (unconstrained) mimizer x_t^* of $Q_{\mu_t}(x)$ that satisfies

$$\|\nabla Q_{\mu_t}(\tilde{x}_t^*)\| \le \tau_t$$

using x_t^{start} as starting point.

Set $t=t+1$, $x_t^{\text{start}} = \tilde{x}_{t-1}^*$

Typical values:

$$\mu_t = c\,\mu_{t-1}, \qquad c = 0.1 \text{ to } 0.5$$

$$\tau_t = c\,\tau_{t-1}$$

The quadratic penalty method

Positive properties of the quadratic penalty method:

•algorithms for unconstrained problems readily available;

•Q is at least as smooth as f, g_i for equality constrained problems;

•usually only very few steps are needed for each penalty parameter, since good starting point known;

•it is not really necessary to solve each unconstrained minimization to high accuracy.

Negative properties of the quadratic penalty method:

•minimizers for finite penalty parameters are usually infeasible;

•problem is becoming more and more ill-conditioned near optimum as penalty parameter is decreased, Hessian large.

The quadratic penalty method

Theorem (Convergence): Let x_t^* be the exact minimizer of $Q_{\mu_t}(x)$ and let $\mu_t \to 0$. Let f,g be once differentiable.

Then every limit point of the sequence $\{x_t^*\}_{t=1,2,\ldots}$ is a solution of the constrained minimization problem

$$\text{minimize}_{x \in D \subset R^n} \quad f(x)$$
$$g(x) = 0$$

The quadratic penalty method

Theorem (Convergence): Let $\tilde{x}_t^*$ be approximate minimizers of $Q_{\mu_t}(x)$ with

$$\|\nabla Q_{\mu_t}(\tilde{x}_t^*)\| \leq \tau_t$$

for a sequence $\tau_t \to 0$ and let $\mu_t \to 0$. Let $f \in C^2, g \in C^{1.}$

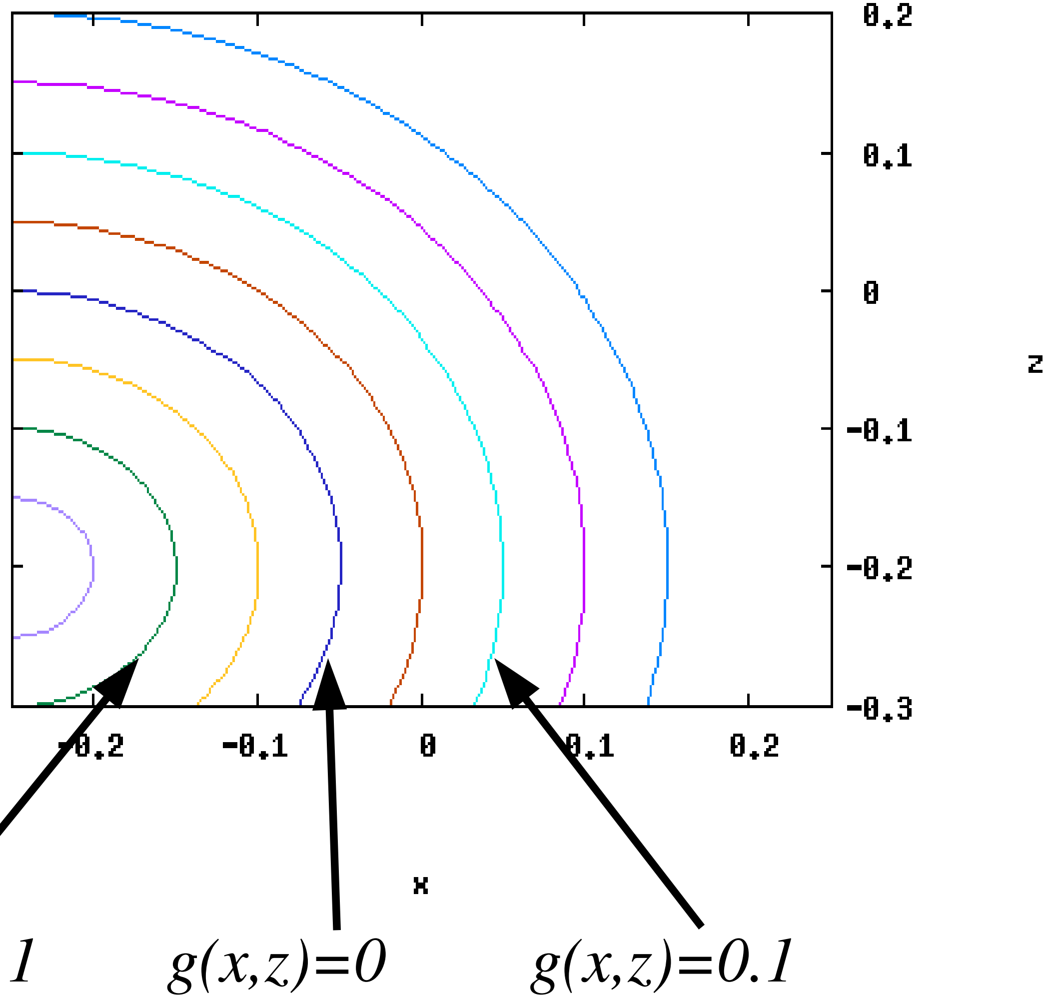
Then every limit point of the sequence $\{x_t^*\}_t = 1,2,\ldots$ satisfies certain first-order necessary conditions for solutions of the constrained minimization problem

$$\text{minimize}_{x \in D \subset R^n} \quad f(x)$$

$$g(x) = 0$$

Lagrange multipliers

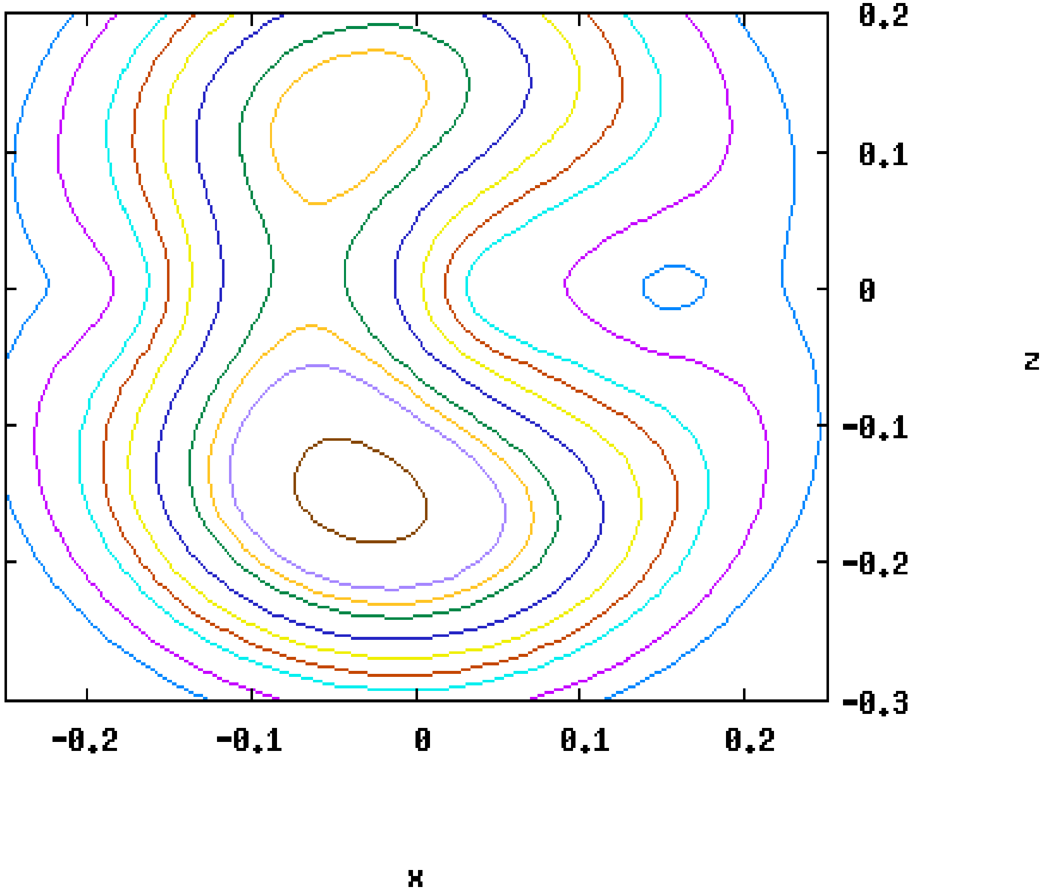
Consider a (single) constraint *g(x)* as a function everywhere:



$$g(\vec{x}) = \|\vec{x} - \vec{x}_0\| - L_{\text{rod}}$$

Lagrange multipliers

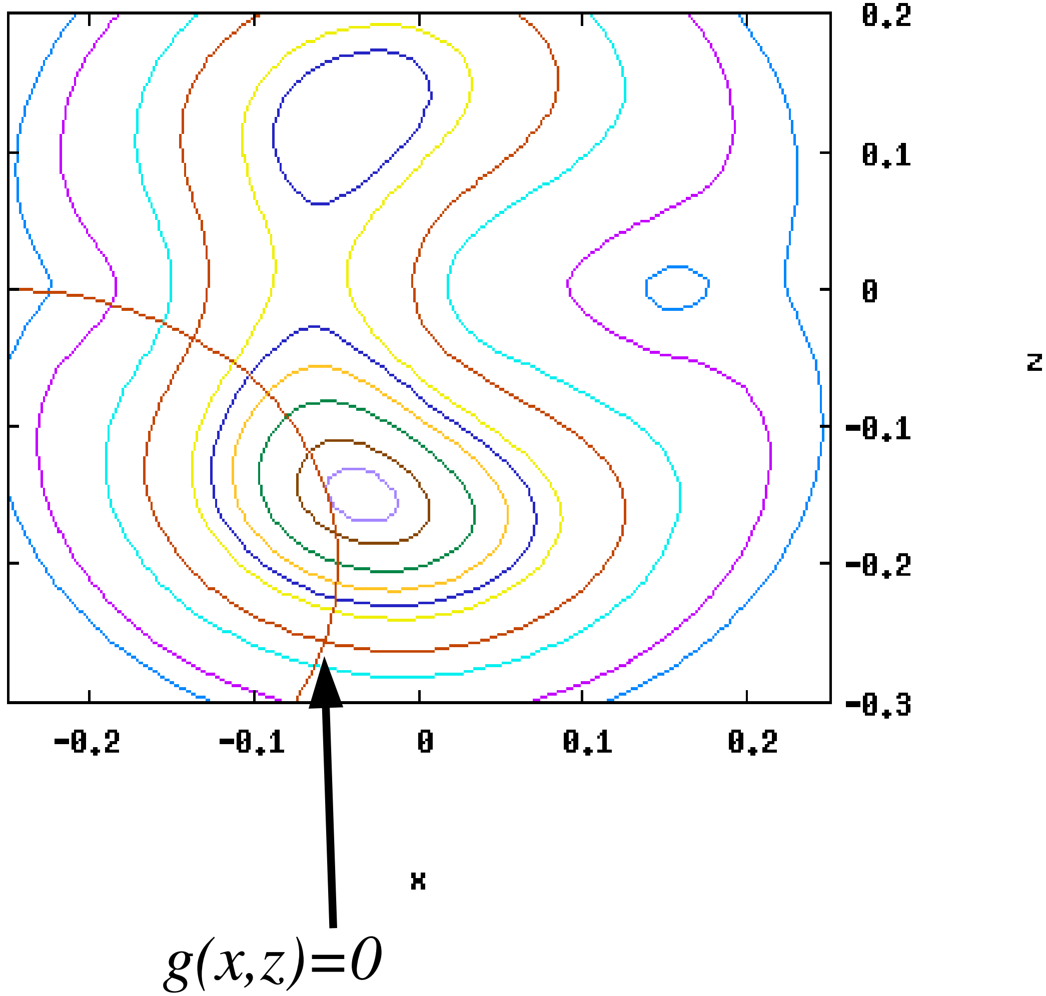
Now look at the objective function *f(x)*:



$$f(\vec{x}) = \sum_{i=1}^{3} \frac{1}{2} D\left(\|\vec{x}-\vec{x}_i\|-L_0\right)^2$$

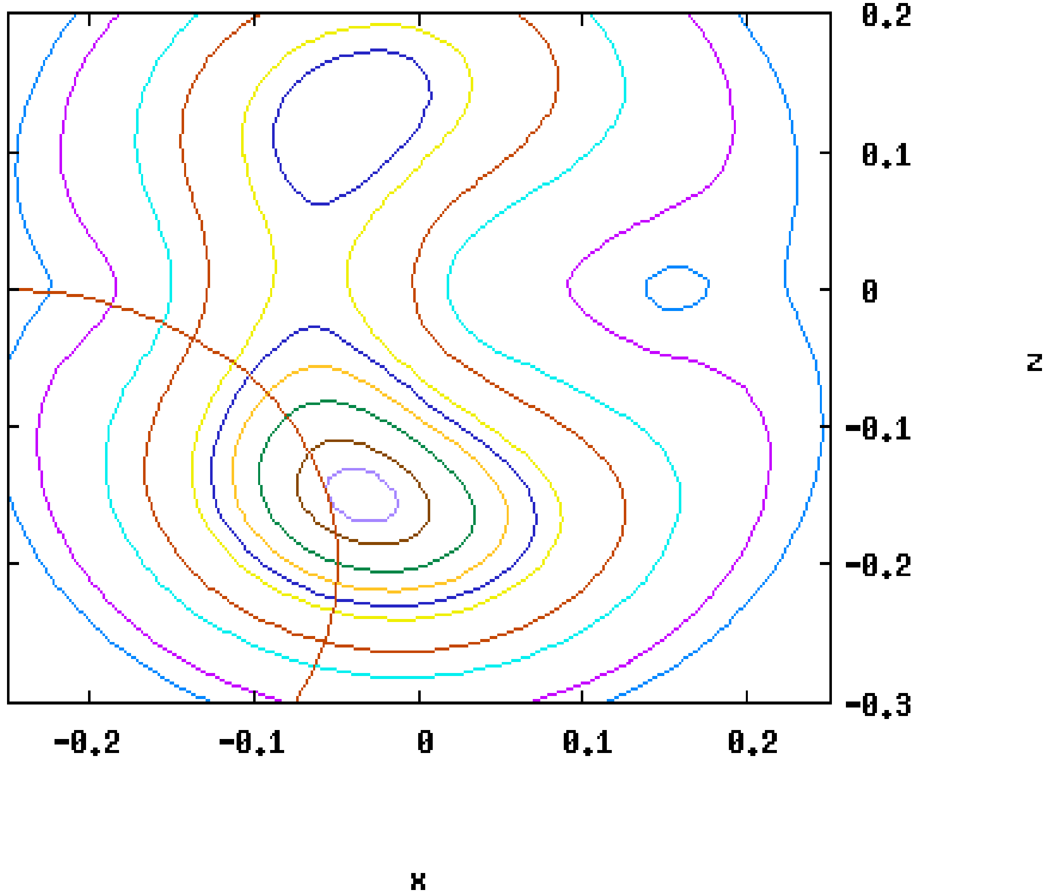
Lagrange multipliers

Now both *f(x)*, *g(x)*:



Lagrange multipliers

Now both *f(x)*, *g(x)*:



Conclusion:

•The solution is where the isocontours are tangential to each other

•The solution is where the gradients of f and g are parallel

•The solution is where $g(x)=0$

Lagrange multipliers

Conclusion:

•The solution is where the gradients of *f* and *g* are parallel

•The solution is where $g(x)=0$

In mathematical terms:

The (local) solutions of

$$\text{minimize} \quad f(\vec{x}) = E(x,z) = \sum_i E_{\text{spring},i}(x,z) + E_{\text{pot}}(x,z)$$

$$g(\vec{x}) = \|\vec{x}-\vec{x}_0\| - L_{\text{rod}} = 0$$

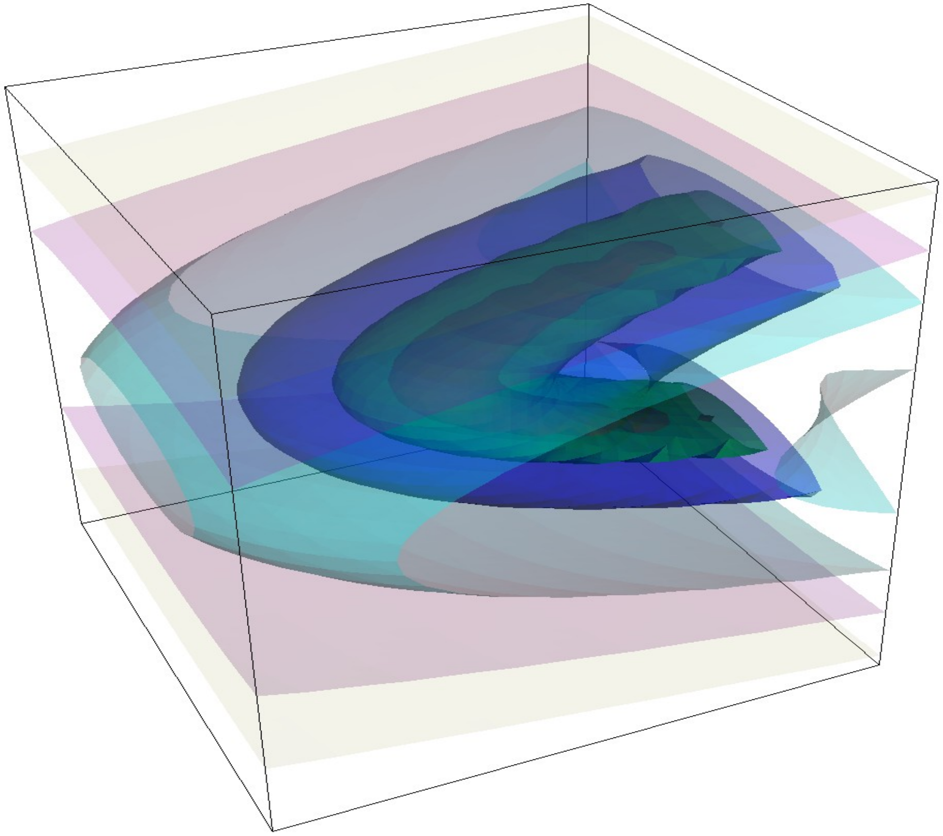
are where the following conditions hold for some value of λ:

$$\nabla f(\vec{x}) - \lambda \nabla g(\vec{x}) = 0$$

$$g(\vec{x}) = 0$$

Lagrange multipliers

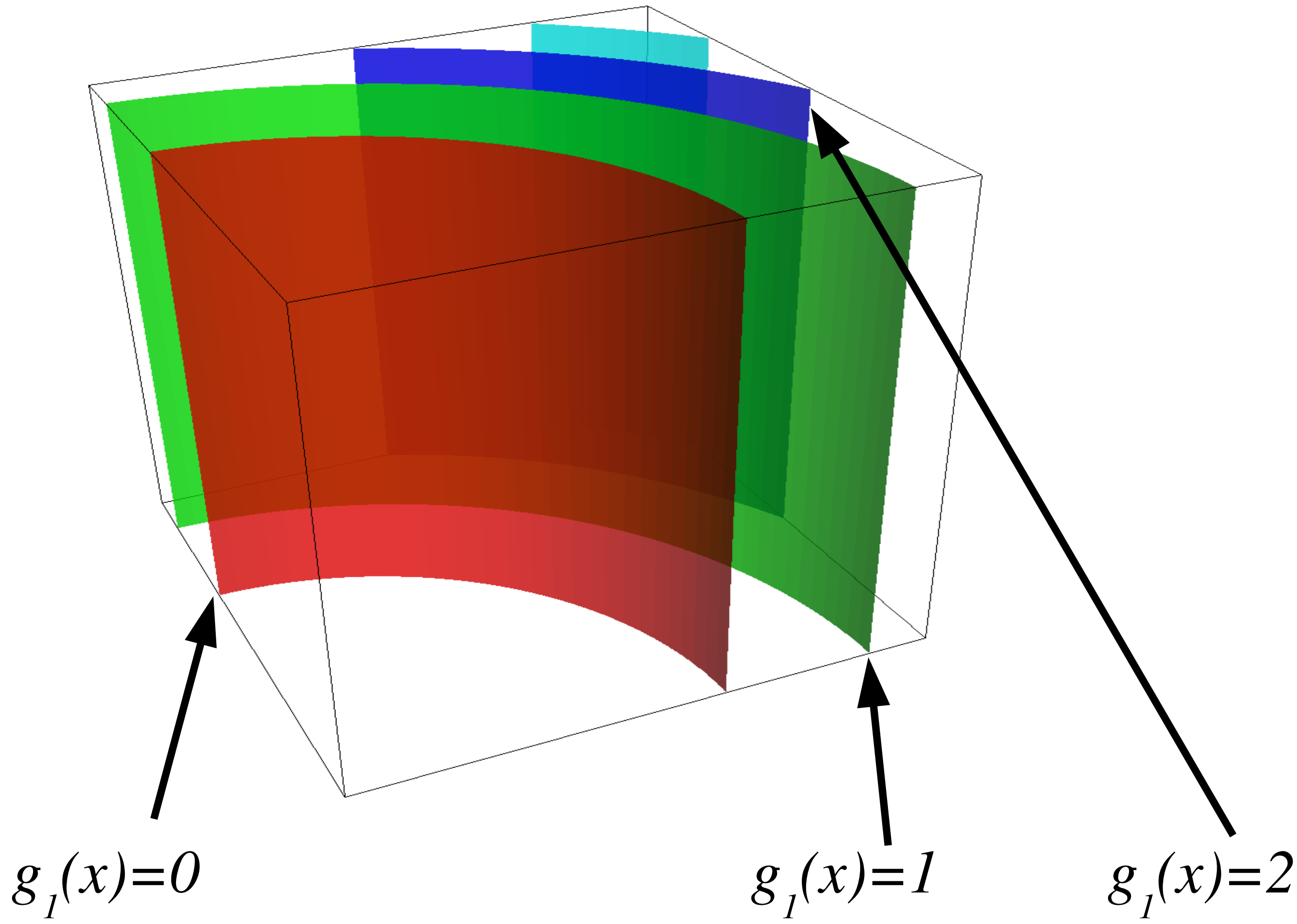
Consider the same situation for three variables and two constraints:



$$f(\vec{x}) = f(x, y, z)$$

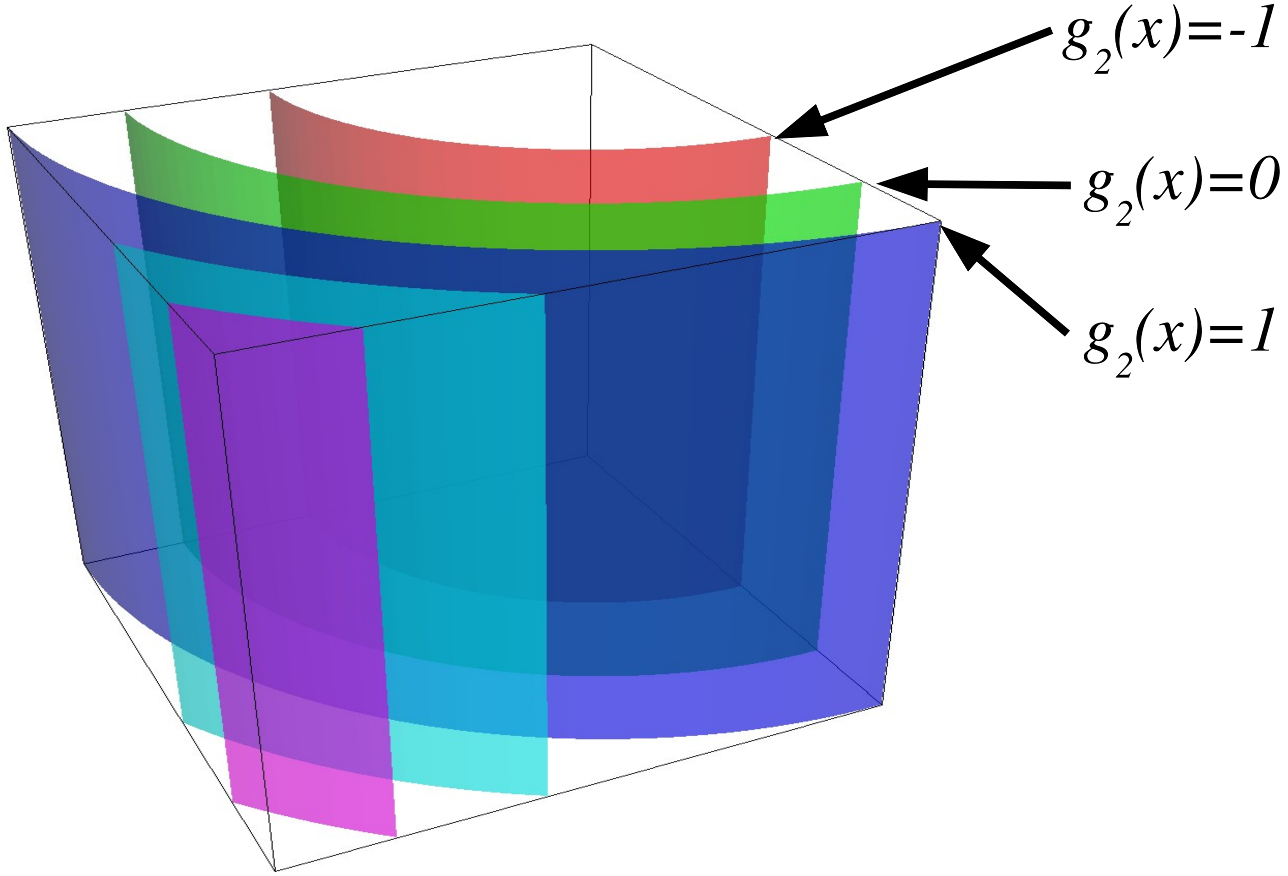
Lagrange multipliers

Constraint 1: Contours of $g_1(x)$



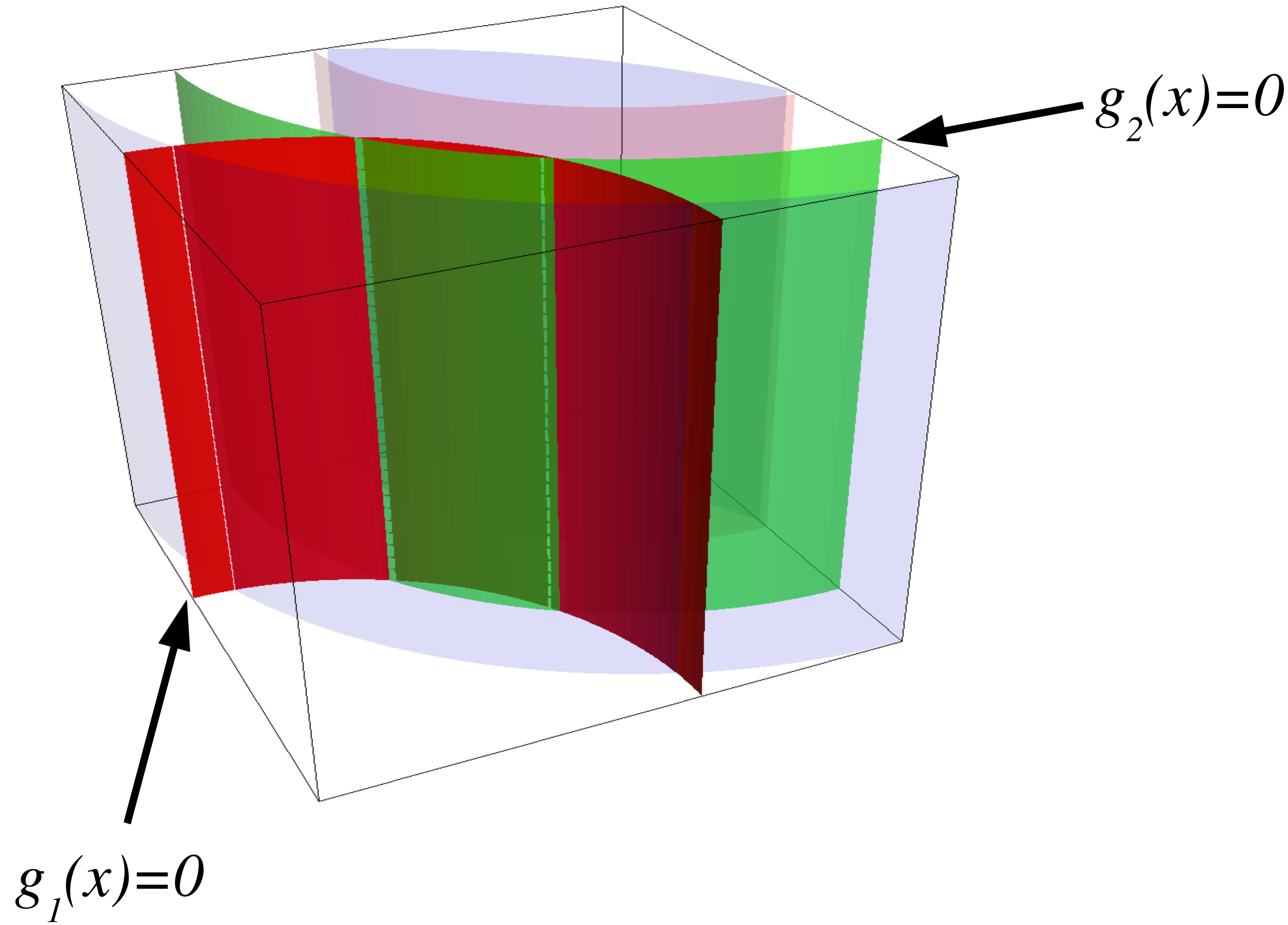
Lagrange multipliers

Constraint 2: Contours of $g_2(x)$



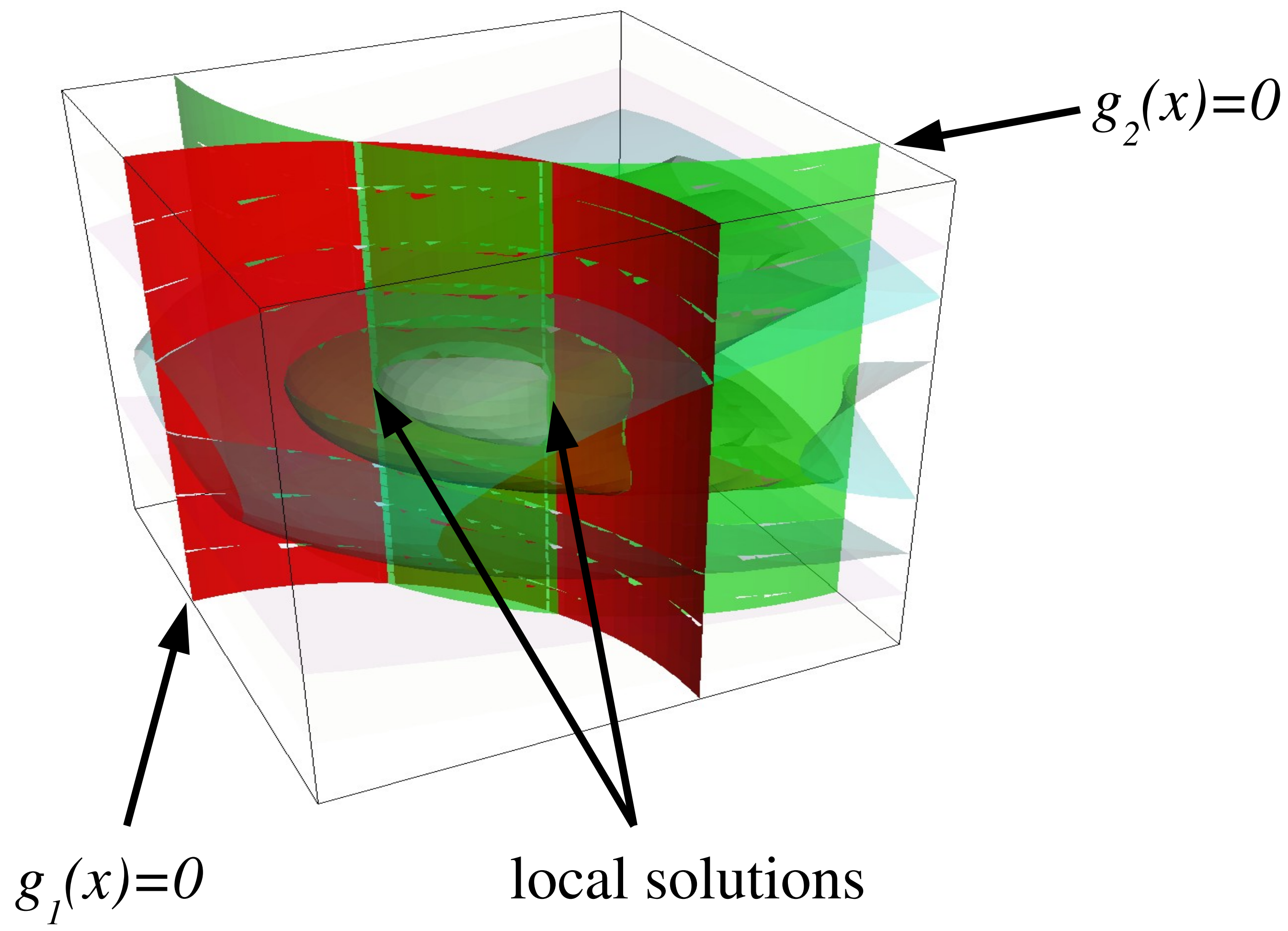
Lagrange multipliers

Constraints 1+2 at the same time

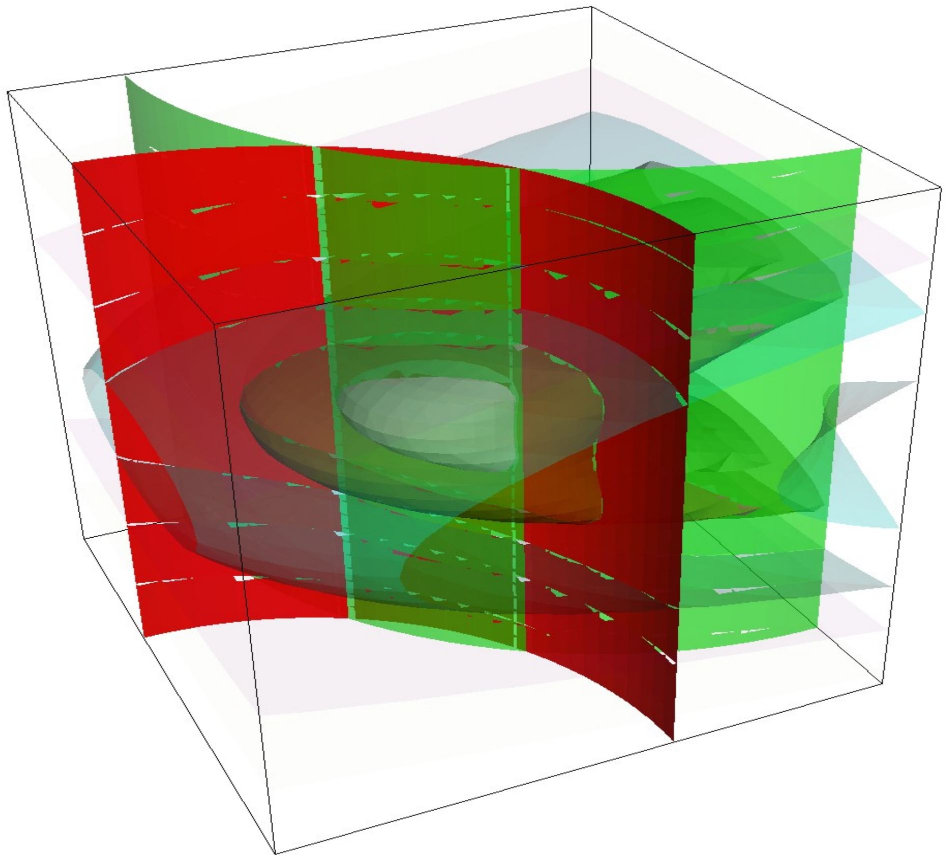


Lagrange multipliers

Constraints 1+2 and *f(x)*:



Lagrange multipliers



Conclusion:

- The solution is where the gradient of f can be written as a linear combination of the gradients of g_1, g_2
- The solution is where $g_1(x)=0$, $g_2(x)=0$

Lagrange multipliers

Generally (under certain conditions):
The (local) solutions of

$$\begin{array}{lll} \text{minimize} & f(\vec{x}) & \qquad f(\vec{x}) : \mathbb{R}^n \rightarrow \mathbb{R} \\ & \vec{g}(\vec{x}) = 0, & \qquad \vec{g}(\vec{x}) : \mathbb{R}^n \rightarrow \mathbb{R}^{n_e} \end{array}$$

are where the conditions

$$\begin{aligned} \nabla f(\vec{x}) - \vec{\lambda} \cdot \nabla \vec{g}(\vec{x}) &= 0 \\ \vec{g}(\vec{x}) &= 0 \end{aligned}$$

hold for some vector of *Lagrange multipliers* $\vec{\lambda} \in \mathbb{R}^{n_e}$

 Note: There are enough equations to determine both x and λ.

Lagrange multipliers

By introducing the *Lagrangian*

$$L(\vec{x},\vec{\lambda}) = f(\vec{x}) - \vec{\lambda}\cdot\vec{g}(\vec{x}), \qquad L : \mathbb{R}^{n}\times\mathbb{R}^{n_e}\rightarrow\mathbb{R}$$

the conditions

$$\begin{aligned} \nabla f(\vec{x}) - \vec{\lambda}\cdot\nabla\vec{g}(\vec{x}) &= 0 \\ \vec{g}(\vec{x}) &= 0 \end{aligned}$$

can conveniently be written as

$$\nabla_{\{\vec{x},\vec{\lambda}\}} L(\vec{x},\vec{\lambda}) = 0$$

Constraint Qualification: Example 1

When is it possible to characterize solutions with Lagrange multipliers?

Consider the problem

$$\begin{aligned} \text{minimize} \quad & f(\vec{x}) = (x+1)^2 + (y+1)^2 + z^{2,} \\ & g_1(\vec{x}) = x = 0, \\ & g_2(\vec{x}) = y = 0. \end{aligned}$$

with solution

$$\vec{x}^* = (0,0,0)^T$$

At the solution, we have

$$\nabla f(\vec{x}^*) = (2,2,0)^T, \quad \nabla g_1(\vec{x}^*) = (1,0,0)^T, \quad \nabla g_2(\vec{x}^*) = (0,1,0)^T$$

and consequently

$$\vec{\lambda} = (2,2)^T$$

Constraint Qualification: Example 1

When is it possible to characterize solutions with Lagrange multipliers?

Compare this with the problem

$$\text{minimize} \quad f(\vec{x}) = (x+1)^2 + (y+1)^2 + z^{2,}$$

$$g_1(\vec{x}) = x^2 = 0,$$

$$g_2(\vec{x}) = y^2 = 0.$$

with the same solution

$$\vec{x}^* = (0,0,0)^T$$

At the solution, we now have

$$\nabla f(\vec{x}^*) = (2,2,0)^T, \quad \nabla g_1(\vec{x}^*) = \nabla g_2(\vec{x}^*) = (0,0,0)^T$$

and there are no Lagrange multipliers so that

$$\nabla f(\vec{x}^*) = \vec{\lambda} \cdot \nabla \vec{g}(\vec{x}^*)$$

Constraint Qualification: Example 2

When is it possible to characterize solutions with Lagrange multipliers?

Consider the problem

$$\begin{aligned} \text{minimize} \quad & f(\vec{x}) = y, \\ & g_1(\vec{x}) = (x-1)^2 + y^2 - 1 = 0, \\ & g_2(\vec{x}) = (x+1)^2 + y^2 - 1 = 0. \end{aligned}$$

There is only a single point at which both constraints are satisfied:

$$\vec{x}^* = (0,0)^T$$

At the solution, we have

$$\nabla f(\vec{x}^*) = (0,1)^T, \quad \nabla g_1(\vec{x}^*) = -\nabla g_2(\vec{x}^*) = (2,0)^T$$

and again there are no Lagrange multipliers so that

$$\nabla f(\vec{x}^*) = \vec{\lambda} \cdot \nabla \vec{g}(\vec{x}^*)$$

Constraint Qualification: LICQ

Definition:
We say that at a point $\vec{x}$ the *linear independence constraint qualification* (LICQ) is satisfied if

$$\{\nabla g_i(\vec{x})\}_{i=1\ldots n_e}$$

is a set of n_e linearly independent vectors.

Note: This is equivalent to saying that the matrix

$$A = \begin{bmatrix} [\nabla g_1(\vec{x})]^T \\ \vdots \\ [\nabla g_{n_e}(\vec{x})]^T \end{bmatrix}$$

has full row rank n_e.

First-order necessary conditions

Theorem:
Suppose that $\vec{x}^*$ is a local solution of

$$\begin{aligned} &\text{minimize} \quad f(\vec{x}) && f(\vec{x}) : \mathbb{R}^n \to \mathbb{R} \\ &\vec{g}(\vec{x}) = 0, && \vec{g}(\vec{x}) : \mathbb{R}^n \to \mathbb{R}^{n_e} \end{aligned}$$

and suppose that at this point the LICQ holds. Then there exists a unique Lagrange multiplier vector so that the following conditions are satisfied:

$$\begin{aligned} \nabla f(\vec{x}) - \vec{\lambda} \cdot \nabla \vec{g}(\vec{x}) &= 0 \\ \vec{g}(\vec{x}) &= 0 \end{aligned}$$

Note: - These conditions are often referred to as the *Karush-Kuhn-Tucker (KKT)* conditions.
- If LICQ does not hold, the problem may still have a solution, but there is no guarantee that it satisfies the KKT conditions!

First-order necessary conditions

Theorem (alternative form):

Suppose that $\vec{x}^*$ is a local solution of

$$\text{minimize} \quad f(\vec{x}) \qquad\qquad f(\vec{x}): \mathbb{R}^n \to \mathbb{R}$$

$$\vec{g}(\vec{x}) = 0, \qquad\qquad \vec{g}(\vec{x}): \mathbb{R}^n \to \mathbb{R}^{n_e}$$

and suppose that at this point the LICQ holds. Then

$$\nabla f(\vec{x}^*)\cdot\vec{w} = 0$$

for every vector tangential to all constraints,

$$\vec{w} \in \left\{\vec{v} : \vec{v}\cdot\nabla g_i(\vec{x}^*) = 0,\ i = 1 \ldots n_e\right\}$$

or equivalently

$$\vec{w} \in \text{Null}(A)$$

Second-order necessary conditions

Theorem:

Suppose that $\vec{x}^*$ is a local solution of

$$\begin{aligned} &\text{minimize} \quad f(\vec{x}) && f(\vec{x}) : \mathbb{R}^n \to \mathbb{R} \\ &\vec{g}(\vec{x}) \; = \; 0, && \vec{g}(\vec{x}) : \mathbb{R}^n \to \mathbb{R}^{n_e} \end{aligned}$$

and suppose that at this point the first order necessary conditions and the LICQ hold. Then

$$\vec{w}^T \nabla^2 f(\vec{x}^*) \cdot \vec{w} \; \geq \; 0$$

for every vector tangential to all constraints,

$$\vec{w} \; \in \; \text{Null}(A)$$

Second-order sufficient conditions

Theorem:

Suppose that at a feasible point $\vec{x}$ the first order necessary (KKT) conditions hold. Suppose also that

$$\vec{w}^T \nabla^2 f(\vec{x}) \cdot \vec{w} \ > \ 0$$

for all tangential vectors

$$\vec{w} \ \in \ \mathrm{Null}(A), \quad \vec{w} \neq 0$$

Then $\vec{x}$ is a strict local minimizer of

$$\begin{array}{lll} \text{minimize} & f(\vec{x}) & f(\vec{x}): \mathbb{R}^n \to \mathbb{R} \\ & \vec{g}(\vec{x}) \ = \ 0, & \vec{g}(\vec{x}): \mathbb{R}^n \to \mathbb{R}^{n_e} \end{array}$$

Characterizing the null space of A

All necessary and sufficient conditions required us to test conditions like

$$\vec{w}^T \nabla^2 f(\vec{x}) \cdot \vec{w} \ > \ 0$$

for all tangential vectors

$$\vec{w} \ \in \ \mathrm{Null}(A), \quad \vec{w} \neq 0$$

In practice, this can be done as follows:

Note that if LICQ holds, then $dim(Null(A))=n\text{-}n_e$. Consequently, there exist $n\text{-}n_e$ vectors z_l so that $Az_l=0$, and every vector w can be written as

$$\vec{w} = Z\vec{\omega}, \quad \vec{w} \in \mathbb{R}^n, \ Z = \left[\vec{z}_1, \ldots, \vec{z}_{n-n_e}\right] \in \mathbb{R}^{n \times n-n_e}, \ \vec{\omega} \in \mathbb{R}^{n-n_e}$$

This matrix Z can be computed from A for example by a QR decomposition.

Characterizing the null space of A

With this matrix Z, the following statements are equivalent:

First order necessary conditions

$$\nabla f(\vec{x}) \cdot \vec{w} = 0 \quad \forall \vec{w} \in \mathrm{Null}(A)$$

$$\left[\nabla f(\vec{x})\right]^T Z = 0$$

Second order necessary conditions

$$\vec{w}^T \nabla^2 f(\vec{x}) \cdot \vec{w} \geq 0 \quad \forall \vec{w} \in \mathrm{Null}(A)$$

$$Z^T \left[\nabla^2 f(\vec{x})\right] Z \text{ is positive semidefinite}$$

Second order sufficient conditions

$$\vec{w}^T \nabla^2 f(\vec{x}) \cdot \vec{w} > 0 \quad \forall \vec{w} \in \mathrm{Null}(A), \quad \vec{w} \neq 0$$

$$Z^T \left[\nabla^2 f(\vec{x})\right] Z \text{ is positive definite}$$

Part 9

Quadratic programming

$$\text{minimize} \quad f(x) = \frac{1}{2} x^T G x + d^T x + e$$
$$g(x) = Ax - b \quad = \quad 0$$

Solving equality constrained problems

Consider a general nonlinear program with general nonlinear equality constraints:

$$\begin{aligned} \text{minimize} \quad & f(x) & \qquad f(x): \mathbb{R}^n \to \mathbb{R} \\ g(x) &= 0, & \qquad g(x): \mathbb{R}^n \to \mathbb{R}^{n_e} \end{aligned}$$

Maybe we can solve such problems with an iterative scheme like unconstrained ones?

Analogy: For unconstrained nonlinear programs, we approximate *f(x)* in each iteration by a quadratic model. For quadratic functions, we can find minima in one step:

$$\min_x f(x) = \frac{1}{2} x^T H x + d^T x + e$$

$$[\nabla^2 f(x_0)] p_0 = -\nabla f(x_0) \;\Leftrightarrow\; x_1 = x_0 - H^{-1}(Hx_0 + d) = -H^{-1} d$$

Solving equality constrained problems

For the general nonlinear constrained problem:

Assuming a condition like LICQ holds, then we know that we need to find points $\{\vec{x}, \vec{\lambda}\}$ at which

$$\begin{aligned} \nabla f(\vec{x}) - \vec{\lambda} \cdot \nabla \vec{g}(\vec{x}) &= 0 \\ \vec{g}(\vec{x}) &= 0 \end{aligned}$$

Alternatively, we can write this as

$$\nabla_{\{\vec{x}, \vec{\lambda}\}} L(\vec{x}, \vec{\lambda}) = 0$$

with

$$L(\vec{x}, \vec{\lambda}) = f(\vec{x}) - \vec{\lambda} \cdot \vec{g}(\vec{x}), \qquad L: \mathbb{R}^n \times \mathbb{R}^{n_e} \rightarrow \mathbb{R}$$

Solving equality constrained problems

If we combine $z = \{x, \lambda\}$ then this can also be written as

$$\nabla_z L(z) = 0$$

which looks exactly like the first-order necessary condition for minimizing the function *L(z)*. We may therefore think of finding solutions for this problem as follows:

- Start at a point $z_0 = [x_0, \lambda_0]^T$
- Compute search directions using $[\nabla_z^2 L(z_k)]\, p_k = -\nabla_z L(z_k)$
- Compute a step length α_k
- Update $z_{k+1} = z_k + \alpha_k p_k$

Note: This is misleading, since we will in fact not look for minima of *L(z)*, but for saddle points. Consequently, $\nabla_z^2 L(z_k)$ is indefinite.

Solving equality constrained problems

The equations we have to solve in each Newton iteration have the form

$$[\nabla_z^2 L(z_k)]\, p_k = -\nabla_z L(z_k)$$

Because

$$L(x,\lambda) = f(x) - \lambda \cdot g(x), \qquad L: \mathbb{R}^n \times \mathbb{R}^{n_e} \to \mathbb{R}$$

the equations we have to solve read in component form:

$$\begin{pmatrix} \nabla^2 f(x_k) - \sum_i \lambda_{i,k} \nabla^2 g_i(x_k) & -\nabla g(x_k) \\ -\nabla g(x_k)^T & 0 \end{pmatrix} \begin{pmatrix} p_k^x \\ p_k^\lambda \end{pmatrix} =$$

$$= -\begin{pmatrix} \nabla f(x_k) - \sum_i \lambda_{i,k} \nabla g_i(x_k) \\ -g(x_k) \end{pmatrix}$$

Linear quadratic programs

Consider first the linear quadratic case with symmetric matrix G:

$$f(x) = \frac{1}{2} x^T G\, x + d^T x + e, \qquad f: \mathbb{R}^n \to \mathbb{R}$$

$$g(x) = Ax - b, \qquad A \in \mathbb{R}^{n_e \times n},\ b \in \mathbb{R}^{n_e}$$

with

$$L(x,\lambda) = f(x) - \lambda^T g(x) = \frac{1}{2} x^T G\, x + d^T x + e - \lambda^T (Ax - b)$$

Then the first search direction needs to satisfy the (linear) set of equations

$$[\nabla_z^2 L(z_0)]\, p_0 = -\nabla_z L(z_0)$$

or equivalently:

$$\begin{pmatrix} G & -A^T \\ -A & 0 \end{pmatrix} \begin{pmatrix} p_0^x \\ p_0^\lambda \end{pmatrix} = -\begin{pmatrix} G x_0 + d - \lambda_0^T A \\ -(A x_0 - b) \end{pmatrix}$$

Linear quadratic programs

Theorem 1: Assume that G is positive definite in all feasible directions, i.e. $Z^T G Z$ is positive definite, and that the matrix A has full row rank. Then the KKT matrix

$$\begin{pmatrix} G & -A^T \\ -A & 0 \end{pmatrix}$$

is nonsingular and the system

$$\begin{pmatrix} G & -A^T \\ -A & 0 \end{pmatrix} \begin{pmatrix} p_0^x \\ p_0^\lambda \end{pmatrix} = -\begin{pmatrix} G x_0 + d - \lambda_0^T A \\ -(A x_0 - b) \end{pmatrix}$$

has a unique solution.

Linear quadratic programs

Theorem 2: Assume that G is positive definite in all feasible directions, i.e. $Z^T G Z$ is positive definite. Then the solution of the linear quadratic program

$$\min_x \quad f(x) = \frac{1}{2} x^T G x + d^T x + e$$
$$g(x) = Ax - b = 0$$

is equivalent to the first iterate

$$x_1 = x_0 + p_0^x$$

that results from solving the linear system

$$\begin{pmatrix} G & -A^T \\ -A & 0 \end{pmatrix} \begin{pmatrix} p_0^x \\ p_0^\lambda \end{pmatrix} = -\begin{pmatrix} G x_0 + d - \lambda_0^T A \\ -(A x_0 - b) \end{pmatrix}$$

irrespective of the starting point x_0.

Linear quadratic programs

Theorem 3: Assume that G is positive definite in all feasible directions, i.e. $Z^T G Z$ is positive definite, and that the matrix A has full row rank. Then the KKT matrix

$$\begin{pmatrix} G & -A^T \\ -A & 0 \end{pmatrix}$$

Has n positive, n_e negative eigenvalues, and no zero eigenvalues. In other words, the KKT matrix is indefinite but non-singular, and the quadratic function

$$L(x,\lambda) = \frac{1}{2} x^T G x + d^T x + e - \lambda^T (Ax - b)$$

in $\{x, \lambda\}$ has a single stationary point that is a saddle point.

Part 10

Sequential Quadratic Programming (SQP)

$$\begin{aligned} \text{minimize} \quad & f(x) \\ & g(x) = 0 \end{aligned}$$

The basic SQP algorithm

For $z = \{x, \lambda\}$ the equality-constrained optimality conditions read

$$\nabla_z L(z) = 0$$

In analogy to Newton's method for unconstrained problems, sequential quadratic programming uses the following basic iteration:

- Start at a point $z_0 = [x_0, \lambda_0]^T$
- Compute search directions using $[\nabla_z^2 L(z_k)]\, p_k = -\nabla_z L(z_k)$
- Compute a step length α_k
- Update $z_{k+1} = z_k + \alpha_k\, p_k$

Computing the SQP search direction

The equations for the search direction are

$$\begin{pmatrix} \nabla^2 f(x_k) - \sum_i \lambda_{i,k} \nabla^2 g_i(x_k) & -\nabla g(x_k) \\ -\nabla g(x_k)^T & 0 \end{pmatrix} \begin{pmatrix} p_k^x \\ p_k^\lambda \end{pmatrix} =$$

$$= -\begin{pmatrix} \nabla f(x_k) - \sum_i \lambda_{i,k} \nabla g_i(x_k) \\ -g(x_k) \end{pmatrix}$$

which we will abbreviate as follows:

$$\begin{pmatrix} W_k & -A_k \\ -A_k^T & 0 \end{pmatrix} \begin{pmatrix} p_k^x \\ p_k^\lambda \end{pmatrix} = -\begin{pmatrix} \nabla f(x_k) - \sum_i \lambda_{i,k} \nabla g_i(x_k) \\ -g(x_k) \end{pmatrix}$$

with

$$W_k = \nabla_x^2 L(x_k, \lambda_k)$$

$$A_k = \nabla_x g(x_k) = -\nabla_x \nabla_\lambda L(x_k, \lambda_x)$$

Computing the SQP search direction

Theorem 1: Assume that W is positive definite in all feasible directions, i.e. $Z_k^T W_k Z_k$ is positive definite, and that the matrix A_k has full row rank. Then the KKT matrix of SQP step k

$$\begin{pmatrix} W_k & -A_k^T \\ -A_k & 0 \end{pmatrix}$$

is nonsingular and the system that determines the SQP search direction

$$\begin{pmatrix} W_k & -A_k^T \\ -A_k & 0 \end{pmatrix}\begin{pmatrix} p_k^x \\ p_k^\lambda \end{pmatrix} = -\begin{pmatrix} \nabla_x L(x_k, \lambda_k) \\ -g(x_k) \end{pmatrix}$$

has a unique solution.

Proof: Use Theorem 1 from Part 9.

Note: The columns of the matrix Z_k span the null space of A_k.

Computing the SQP search direction

Theorem 2: The solution of the SQP search direction system

$$\begin{pmatrix} W_k & -A_k^T \\ -A_k & 0 \end{pmatrix} \begin{pmatrix} p_k^x \\ p_k^\lambda \end{pmatrix} = -\begin{pmatrix} \nabla_x L(x_k, \lambda_k) \\ -g(x_k) \end{pmatrix}$$

equals the minimizer of the problem

$$\min_x \; m_k(p_k^x) = L(x_k, \lambda_k) + \nabla_x L(x_k, \lambda_k)^T p_k^x + \frac{1}{2} p_k^{xT} \nabla_x^2 L(x_k, \lambda_k) p_k^x$$

$$g(x_k) + \nabla g(x_k)^T p_k^x = 0$$

that approximates the original nonlinear equality-constrained minimization problem.

Proof: Essentially just use Theorem 2 from Part 9.

Note: This means that SQP in each step minimizes a quadratic model of the Lagrangian, subject to linearized constraints.

Computing the SQP search direction

Theorem 3: The SQP iteration with full steps, i.e.

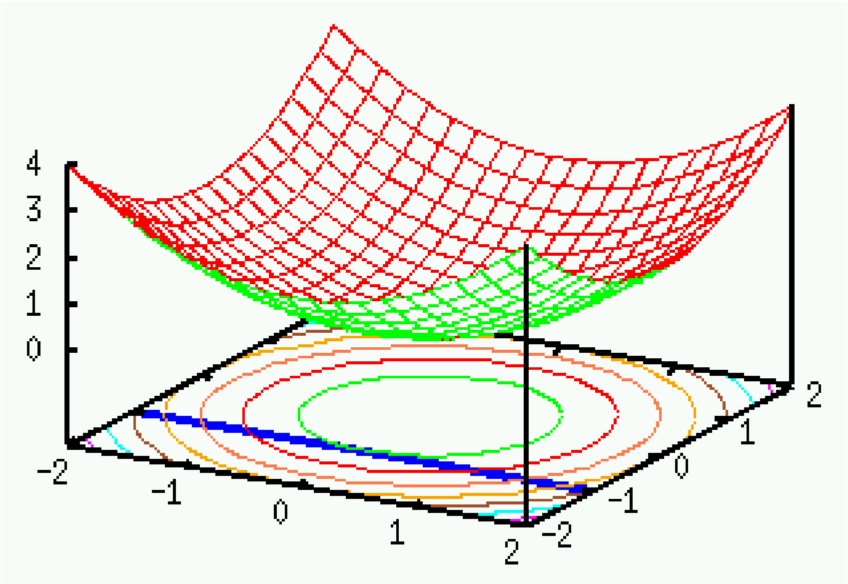
$$\begin{pmatrix} W_k & -A_k^T \\ -A_k & 0 \end{pmatrix} \begin{pmatrix} p_k^x \\ p_k^\lambda \end{pmatrix} = -\begin{pmatrix} \nabla_x L(x_k, \lambda_k) \\ -g(x_k) \end{pmatrix}$$

$$x_{k+1} = x_k + p_k^x, \qquad \lambda_{k+1} = \lambda_k + p_k^\lambda$$

converges to the solution of the constrained nonlinear optimization problem with *quadratic order* if (i) we start close enough to the solution, (ii) the LICQ holds at the solution and (iii) the matrix $Z_*^T W_* Z_*$ is positive definite at the solution.

How SQP works

Example 1:



$$\min f(x) = \frac{1}{2}(x_1^2 + x_2^2)$$

$$g(x) = x_2 + 1 = 0$$

The search direction is then computed using the step

$$\min \; m_k(p_k^x) = \frac{1}{2} p_k^{xT} p_k^x + x_k^T p_k^x + \begin{pmatrix} 0 \\ 1 \end{pmatrix}^T p_k^x + L(x_k, \lambda_k)$$

$$x_{2,k} + 1 + \begin{pmatrix} 0 \\ 1 \end{pmatrix}^T p_k^x = 0$$

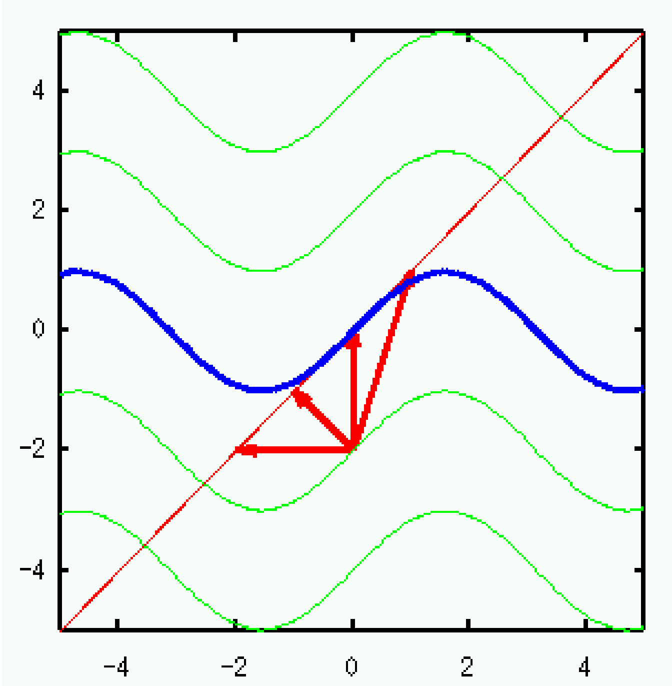
In other words, the linearized constraint enforces that

$$p_{2,k}^x = -(x_{2,k} - 1) \quad \rightarrow \quad x_{2,k+1} = x_{2,k} + p_{2,k}^x = -1$$

How SQP works

Example 2:

$$\min\ f(x)$$
$$g(x) = x_2 - \sin(x_1) = 0$$



The search direction is then computed by

$$\min\ m_k(p_k^x)$$
$$x_{2,k} - \sin(x_{1,k}) + \begin{pmatrix} -\cos(x_{1,k}) \\ 1 \end{pmatrix}^T p_k^x = 0$$

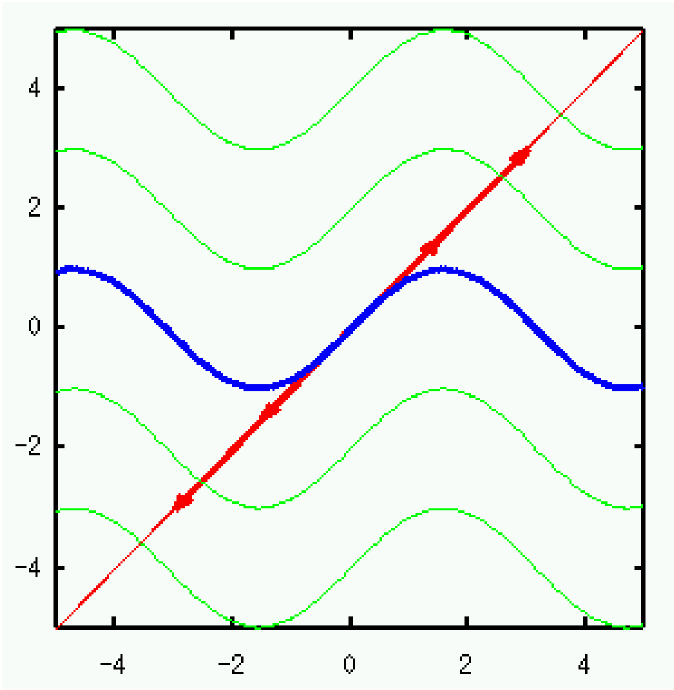
In particular, if we are currently at *(0,-2)*, this enforces

$$-p_{1,k} + p_{2,k} = 2$$

How SQP works

Example 3:

$$\min\ f(x)$$
$$g(x) = 0$$



If the constraint is already satisfied at a step, then search direction is computed by

$$\min\ m_k(p_k^x)$$
$$g(x_k) + \nabla g(x_k)^T p_k^x = \nabla g(x_k)^T p_k^x = 0$$

In other words: The update step can only be tangential to the constraint (along the linearized constraint)!

Hessian modifications for SQP

The SQP step

$$\begin{pmatrix} W_k & -A_k^T \\ -A_k & 0 \end{pmatrix} \begin{pmatrix} p_k^x \\ p_k^\lambda \end{pmatrix} = -\begin{pmatrix} \nabla_x L(x_k,\lambda_k) \\ -g(x_k) \end{pmatrix}$$

is equivalent to the minimization problem

$$\min_x \; m_k(p_k^x) = L(x_k,\lambda_k) + \nabla_x L(x_k,\lambda_k)^T p_k^x + \frac{1}{2} p_k^{xT} \nabla_x^2 L(x_k,\lambda_k)\, p_k^x$$

$$g(x_k) + \nabla g(x_k)^T p_k^x = 0$$

or abbreviated:

$$\min_x \; m_k(p_k^x) = L_k + (\nabla_x f_k^T - \lambda_k^T A_k)\, p_k^x + \frac{1}{2} p_k^{xT} W_k\, p_k^x$$

$$g(x_k) + A_k^T p_k^x = 0$$

From this, we may expect to get into trouble if the matrix $Z_k^T W_k Z_k$ is not positive definite.

Hessian modifications for SQP

If the matrix $Z_k^T W_k Z_k$ in the SQP step

$$\begin{pmatrix} W_k & -A_k^T \\ -A_k & 0 \end{pmatrix} \begin{pmatrix} p_k^x \\ p_k^\lambda \end{pmatrix} = -\begin{pmatrix} \nabla_x L(x_k, \lambda_k) \\ -g(x_k) \end{pmatrix}$$

is not positive definite, then there may not be a unique solution.

There exist a number of modifications to ensure that an alternative step can be computed that satisfies

$$\begin{pmatrix} \tilde{W}_k & -A_k^T \\ -A_k & 0 \end{pmatrix} \begin{pmatrix} p_k^x \\ p_k^\lambda \end{pmatrix} = -\begin{pmatrix} \nabla_x L(x_k, \lambda_k) \\ -g(x_k) \end{pmatrix}$$

instead.

Line search procedures for SQP

Motivation: For unconstrained problems, we could look to $f(x)$ to measure progress along a search direction p_k computed from a quadratic model m_k that approximates $f(x)$.

Idea: For constrained problems, we could think of using $L(z)$ to measure progress along a search direction p_k computed using the SQP step based on the model m_k.

Problem 1: The Lagrangian $L(z)$ is unbounded. For example, for linear-quadratic problems, it is a quadratic function of saddle-point form. Instead of a minimum we are now looking for this saddle point of L.

Consequence 1: We can't use $L(z)$ to measure progress in a line search algorithms using, for example, the Wolfe conditions.

Line search procedures for SQP

Motivation: For unconstrained problems, we could look to $f(x)$ to measure progress along a search direction p_k computed from a quadratic model m_k that approximates $f(x)$.

Idea: For constrained problems, we could think of using $L(z)$ to measure progress along a search direction p_k computed using the SQP step based on the model m_k.

Problem 2: Some step lengths may lead to a significant reduction in $f(x)$ but take us far away from the constraints $g(x)=0$. Is this better that a step that may *increase* $f(x)$ but lands *on the constraint* ?

Consequence 2: We need a *merit function* that balances our desire to decrease $f(x)$ while satisfying the constraint $g(x)$.

Line search procedures for SQP

Solution: Drive step length determination using a *merit* function that contains both *f(x)* and *g(x)*.

Examples: The most commonly used choices are to use either the l_1 merit function

$$\phi_1(x) = f(x) + \frac{1}{\mu}\|g(x)\|_1$$

with

$$\frac{1}{\mu} = \|\lambda_{k+1}\|_\infty + \bar{\delta}, \qquad \bar{\delta} > 0$$

or *Fletcher's* merit function

$$\phi_F(x) = f(x) - \lambda(x)^T g(x) + \frac{1}{2\mu}\|g(x)\|^2$$

with

$$\lambda(x) = [A(x)A(x)^T]^{-1} A(x)\nabla f(x)$$

Line search procedures for SQP

Definition: A merit functions is called *exact* if the constrained optimizer of the problem

$$\min_x \; f(x)$$
$$g(x) = 0$$

is also a minimizer of the merit function.

Note: Both the l_1 and Fletcher's merit function

$$\phi_1(x) = f(x) + \frac{1}{\mu}\|g(x)\|_1$$
$$\phi_F(x) = f(x) - \lambda(x)^T g(x) + \frac{1}{2\mu}\|g(x)\|^2$$

are exact for appropriate choices of λ, μ .

Line search procedures for SQP

Theorem 4: The SQP search direction that satisfies

$$\begin{pmatrix} W_k & -A_k^T \\ -A_k & 0 \end{pmatrix} \begin{pmatrix} p_k^x \\ p_k^\lambda \end{pmatrix} = -\begin{pmatrix} \nabla_x L(x_k, \lambda_k) \\ -g(x_k) \end{pmatrix}$$

is a direction of descent for both the l_1 merit function as well as Fletcher's merit function if (i) the current point x_k is not a stationary point of the equality-constrained problem, and (ii) the matrix $Z_k^T W_k Z_k$ is positive definite.

A practical SQP algorithm

Algorithm: For $k=0,1,2,\ldots$

•Find an update using the KKT system

$$\begin{pmatrix} W_k & -A_k^T \\ -A_k & 0 \end{pmatrix} \begin{pmatrix} p_k^x \\ p_k^\lambda \end{pmatrix} = -\begin{pmatrix} \nabla_x L(x_k, \lambda_k) \\ -g(x_k) \end{pmatrix}$$

•Determine a step length using a backtracking linear search, a merit function and the Wolfe (or Goldstein) conditions:

$$\phi(x_k + \alpha p_k^x) \le \phi(x_k) + c_1 \alpha \nabla \phi(x_k) \cdot p_k^x$$

$$\nabla \phi(x_k + \alpha p_k^x) \cdot p_k^x \ge c_2 \nabla \phi(x_k) \cdot p_k^x$$

•Update the iterate using either

$$x_{k+1} = x_k + \alpha_k p_k^x, \qquad \lambda_{k+1} = \lambda_k + \alpha_k p_k^\lambda$$

or

$$x_{k+1} = x_k + \alpha_k p_k^x, \qquad \lambda_{k+1} = [A_{k+1} A_{k+1}^T]^{-1} A_{k+1} \nabla f(x_{k+1})$$

Parts 8-10

Summary of methods for equality-constrained Problems

$$\begin{aligned} \text{minimize} \quad & f(x) \\ & g_i(x) = 0, \qquad i = 1, \ldots, n_e \end{aligned}$$

Summary of methods

There are two general methods for equality-constrained problems:

- Penalty methods (e.g. the quadratic penalty method) convert the constrained problem into an unconstrained one that can be solved with the techniques we already know.

  However, they often lead to ill-conditioned problems

- Lagrange multipliers allow to reformulate the problem into one where we look for saddle points of a Lagrangian

- Sequential quadratic programming (SQP) methods look for these saddle points by solving a sequence of quadratic programs with linear constraints, which are simple to solve

- SQP methods are the most powerful methods to solve equality-constrained problems efficiently.

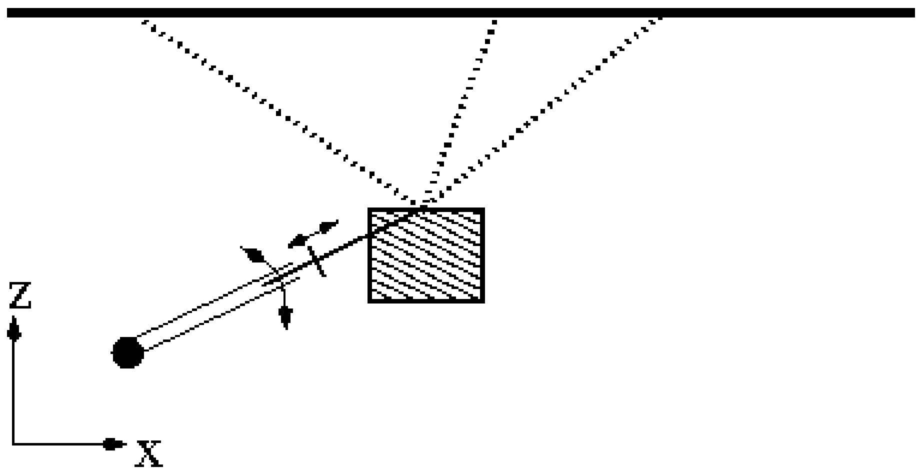
Part 11

Inequality-constrained Problems

$$
\begin{aligned}
\text{minimize} \quad & f(x) \\
& g_i(x) = 0, \qquad i = 1, \ldots, n_e \\
& h_i(x) \geq 0, \qquad i = 1, \ldots, n_i
\end{aligned}
$$

An example

Consider the example of the body suspended from a ceiling with springs, but with an element of fixed *minimal* length attached to a fixed point:



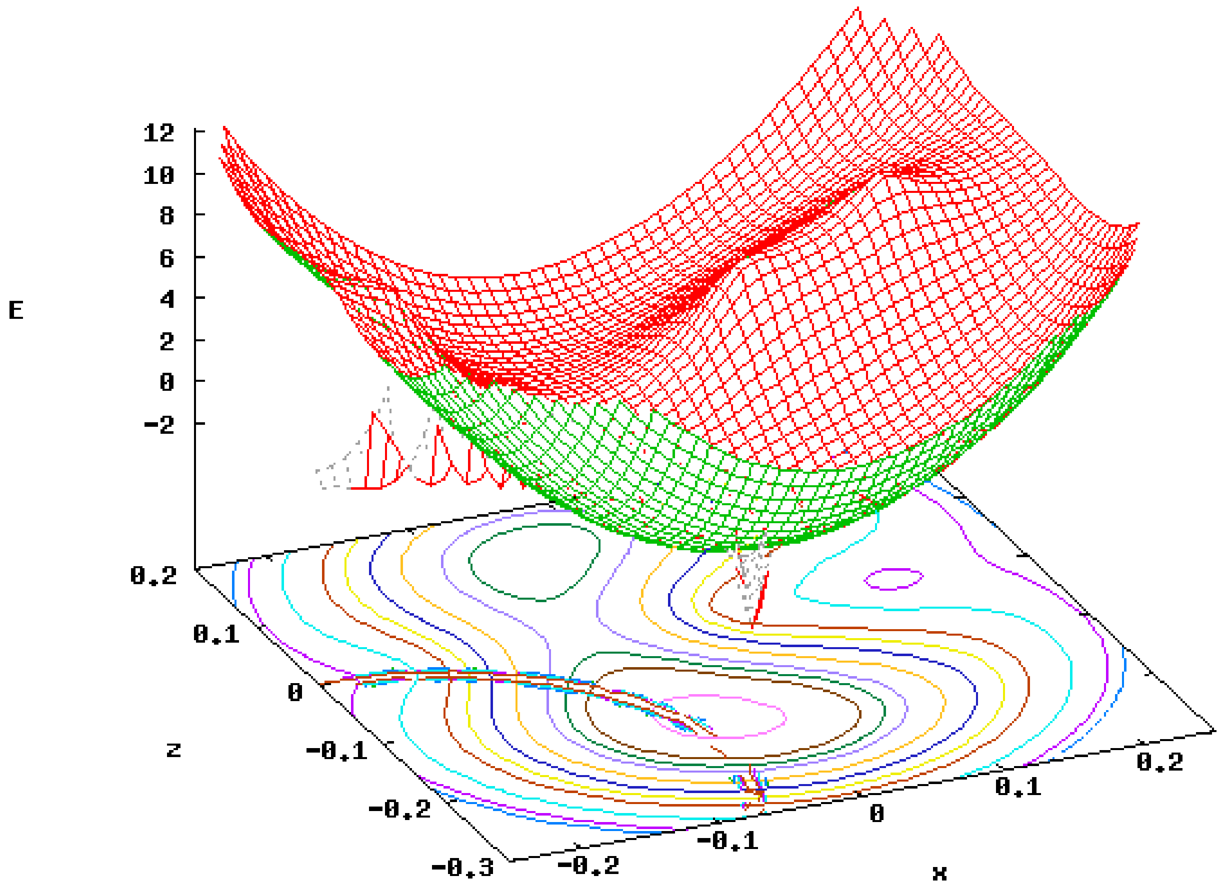
To find the position of the body we now need to solve the following problem:

$$\text{minimize} \quad f(\vec{x}) = E(x,z) = \sum_i E_{\text{spring},i}(x,z) + E_{\text{pot}}(x,z)$$

$$\|\vec{x} - \vec{x}_0\| - L_{\text{rod}} \;\; \geq \;\; 0$$

An example

We can gain some insight into the problem by plotting the energy as a function of *(x,z)* along with the constraint:



Definitions

We call this the standard form of inequality constrained problems:

$$
\begin{aligned}
\text{minimize}_{x \in D \subset R^n} \quad & f(x) \\
& g_i(x) = 0, \quad i = 1 \ldots n_e \\
& h_i(x) \geq 0, \quad i = 1 \ldots n_i
\end{aligned}
$$

We will also frequently write this as follows, implying (in)equality elementwise:

$$
\begin{aligned}
\text{minimize}_{x \in D \subset R^n} \quad & f(x) \\
& g(x) = 0 \\
& h(x) \geq 0
\end{aligned}
$$

Definitions

Let x^* be the solution of

$$\begin{aligned} \text{minimize}_{x\in D\subset R^n} \quad & f(x) \\ g_i(x) &= 0, \quad i=1\ldots n_e \\ h_i(x) &\geq 0, \quad i=1\ldots n_i \end{aligned}$$

We call a constraint *active* if it is zero at the solution x^*:

- Obviously, all equality constraints are active, since a solution needs to satisfy $g(x^*)=0$
- Some inequality constraints may not be active if it so happens that $h_i(x^*)>0$ for some index i
- Other inequality constraints may be active if $h_i(x^*)=0$

We call the set of all active (equality and inequality) constraints the *active set.*

Definitions

Note: If x^* is the solution of

$$\begin{aligned}
\text{minimize}_{x\in D\subset R^n} \quad & f(x) \\
g_i(x) &= 0, \quad i=1\ldots n_e \\
h_i(x) &\geq 0, \quad i=1\ldots n_i
\end{aligned}$$

then it is also the solution of the problem

$$\begin{aligned}
\text{minimize}_{x\in D\subset R^n} \quad & f(x) \\
g_i(x) &= 0, \quad i=1\ldots n_e \\
h_i(x) &= 0, \quad i=1\ldots n_i, i \text{ is active at } x^*
\end{aligned}$$

where we have dropped all *inactive* constraints and made equalities out of all active constraints.

Definitions

A trivial reformulation of the problem is obtained by defining the *feasible set*:

$$\Omega = \{ x \in R^n : \ g(x) = 0, h(x) \geq 0 \}$$

Then the original problem is equivalently recast as

$$\text{minimize}_{x \in D \cap \Omega \subset R^n} \ f(x)$$

Note 1: This reformulation is not of much practical interest.

Note 2: The feasible set can be continuous or discrete. It can also be empty if the constraints are mutually incompatible. In the following we will always assume that it is continuous and non-empty.

The quadratic penalty method

Observation: The solution of

$$\text{minimize}_{x\in D\subset R^n} \quad f(x)$$
$$g(x) = 0$$
$$h(x) \geq 0$$

must lie within the feasible set.

Idea: Let's *relax* the constraint and allow to search also in where *g(x)* is small but not zero, or where *h(x)* is small and negative. However, make sure that the objective function becomes very large if far away from the feasible set:

$$\text{minimize}_{x\in D\subset R^n} \quad Q_\mu(x) = f(x) + \frac{1}{2\mu}\|g(x)\|^2 + \frac{1}{2\mu}\|[h(x)]^-\|^2$$

$Q_\mu(x)$ is called the *quadratic relaxation* of the minimization problem. μ is the *penalty parameter*, and

$$[h(x)]^- = \min\{0, h(x)\}$$

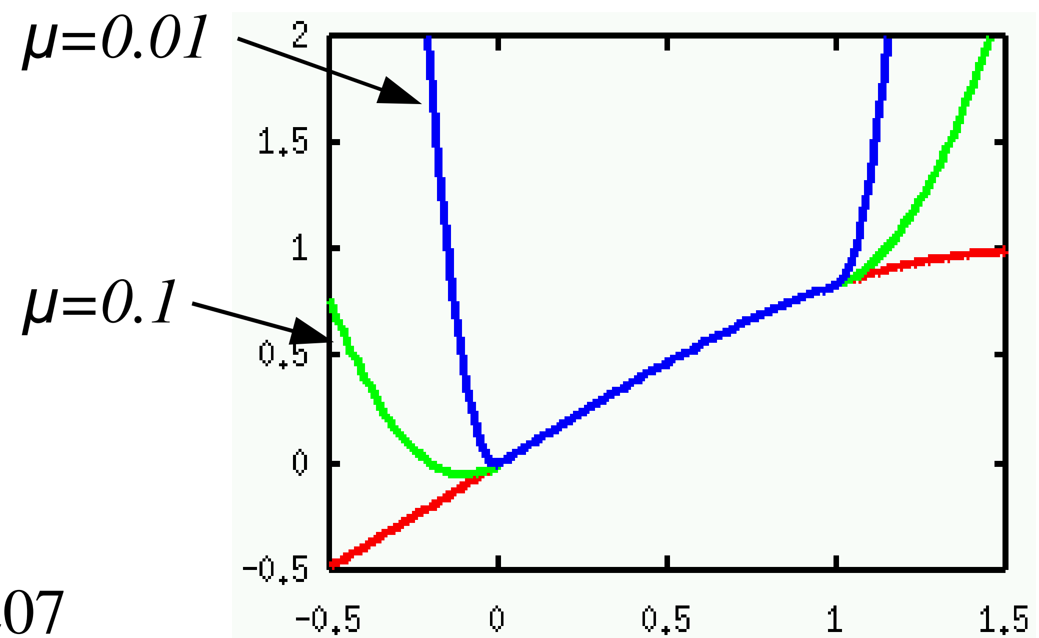
The quadratic penalty method

Replace the original constrained minimization problem

$$\begin{aligned} \text{minimize} \quad & f(x) \\ & g_i(x) = 0, \qquad i=1,\dots,n_e \\ & h_i(x) \geq 0, \qquad i=1,\dots,n_i \end{aligned}$$

by an unconstrained method with a quadratic penalty term:

$$\text{minimize}_{x \in D \subset R^n} \quad Q_\mu(x) = f(x) + \frac{1}{2\mu}\|g(x)\|^2 + \frac{1}{2\mu}\|[h(x)]^-\|^2$$

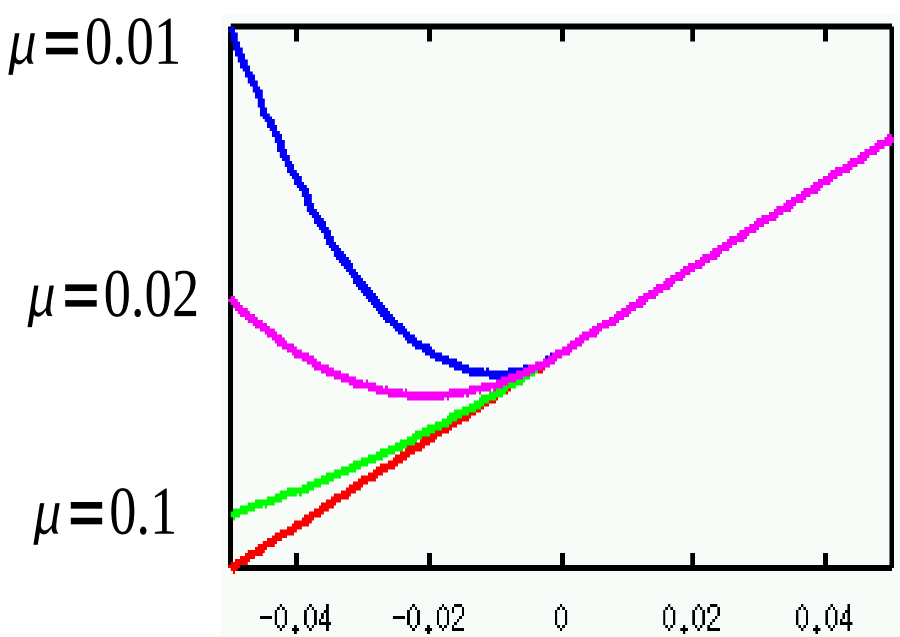


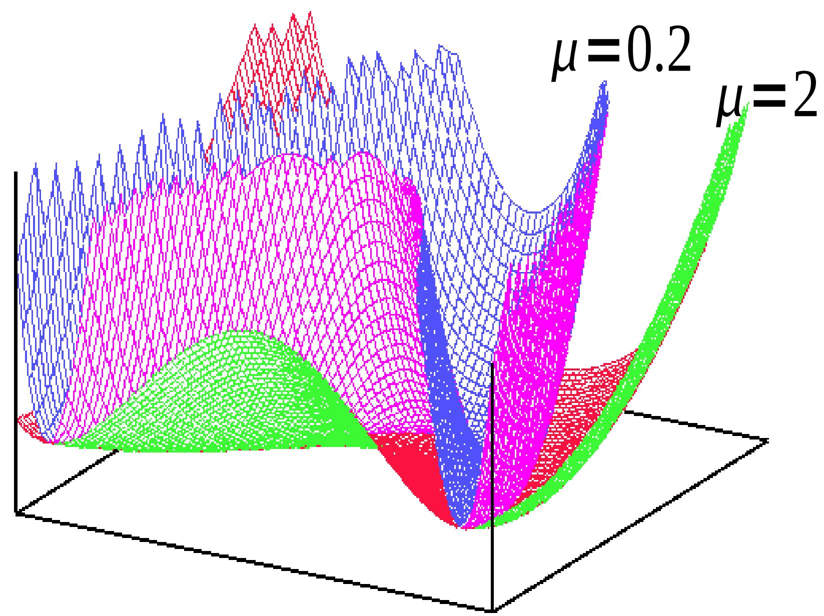
Example:

$$\begin{aligned} \text{minimize} \quad & f(x) = \sin(x) \\ & h_1(x) = x - 0 \geq 0, \\ & h_2(x) = 1 - x \geq 0. \end{aligned}$$

Negative properties of the quadratic penalty method:

- minimizers for finite penalty parameters are usually *infeasible*;
- problem is becoming more and more ill-conditioned near optimum as penalty parameter is decreased, Hessian large;
- for inequality constrained problems, Hessian not twice differentiable at constraints.





minimize x_2^2 $\quad$ $s.t.\, g(x)=x_2+x_1^2=0$

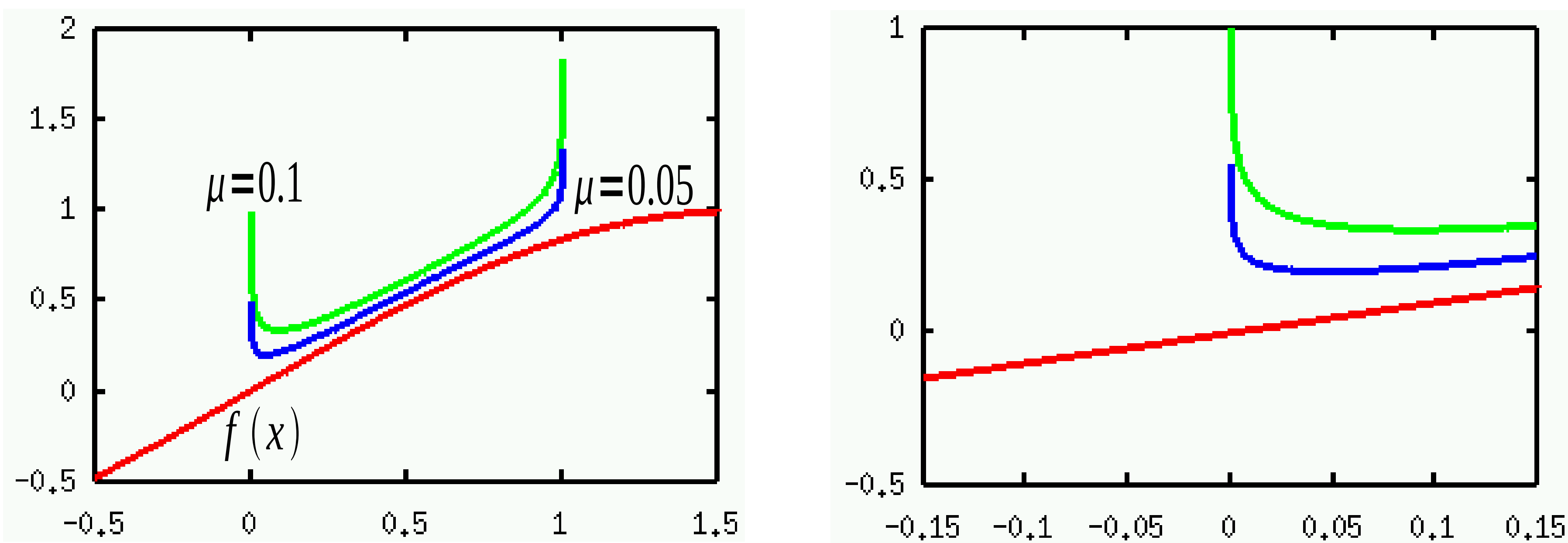
The logarithmic barrier method

Replace the original constrained minimization problem

$$\begin{aligned} &\text{minimize} \quad f(x) \\ &\qquad\qquad\quad h_i(x) \;\geq\; 0, \qquad i=1,\ldots,n_i \end{aligned}$$

by an unconstrained method with a logarithmic *barrier* term:

$$\text{minimize}_{x\in D\subset R^n} \quad Q_\mu(x) = f(x) + \mu \sum_{i=1}^{n_i} -\log h_i(x)$$



$$\text{minimize} \quad f(x) = \sin(x) \qquad \text{s.t.} \quad x \geq 0, \quad x \leq 1$$

The logarithmic barrier method

Properties of successive minimization of

$$\text{minimize}_x \quad Q_\mu(x) = f(x) - \mu \sum_i \log h_i(x)$$

- intermediate minimizers are feasible, since $Q_\mu(x)=\infty$ in the infeasible region; the method is an *interior point method*.
- Q is smooth if constraints are smooth;
- we need a feasible point as starting point;
- ill-conditioning and inadequacy of Taylor expansion remain;
- $Q_\mu(x)$ may be unbounded from below if $h(x)$ unbounded.
- inclusion of equality constraints as before by quadratic penalty method.

Summary:

This is an efficient method for the solution of constrained problems.

Algorithms for penalty/barrier methods

Algorithm (exactly as for the equality constrained case):

Given x_0^{start}, $\{\mu_t\} \to 0$, $\{\tau_t\} \to 0$

For t=0,1, 2, ...:

Find an approximation $\tilde{x}_t^*$ to the (unconstrained) mimizer x_t^* of $Q_{\mu_t}(x)$ that satisfies

$$\|\nabla Q_{\mu_t}(\tilde{x}_t^*)\| \le \tau_t$$

using x_t^{start} as starting point.

Set $t=t+1$, $x_t^{\text{start}} = \tilde{x}_{t-1}^*$

Typical values:

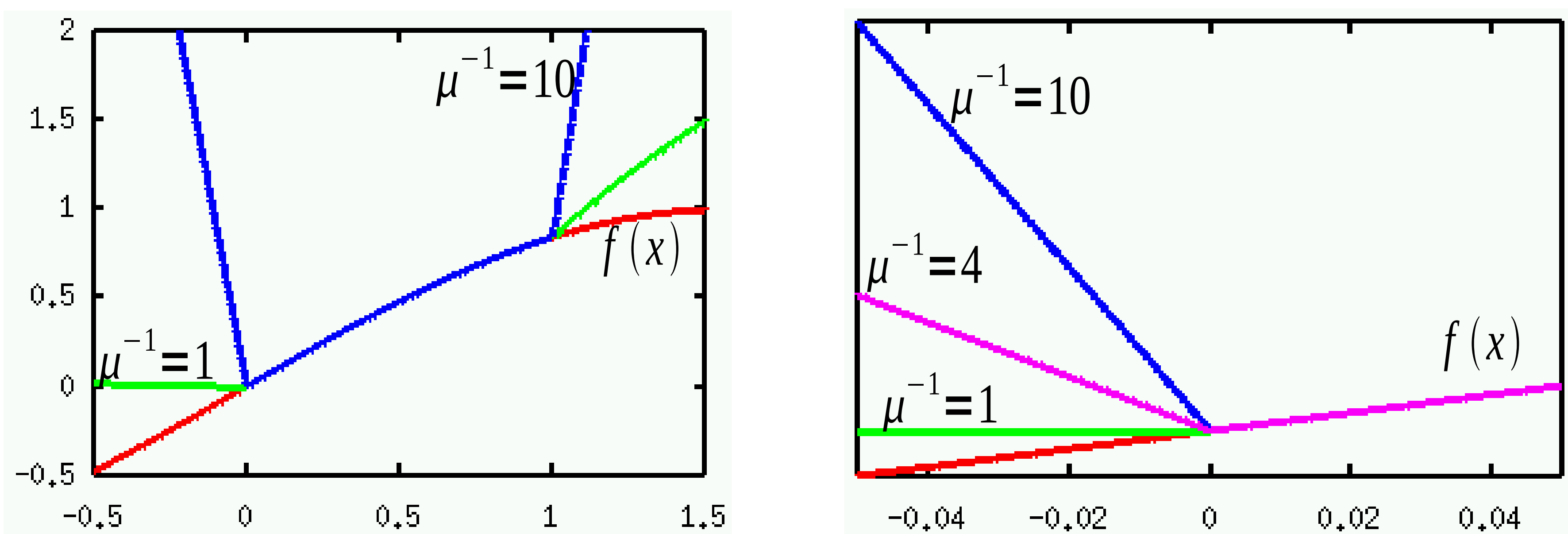
$$\mu_t = c\,\mu_{t-1}, \qquad c = 0.1 \text{ to } 0.5$$

$$\tau_t = c\,\tau_{t-1}$$

The exact penalty method

Previous methods suffered from the fact that minimizers of $Q_\mu(\boldsymbol{x})$ for finite μ are not optima of the original problem. Solution: use

$$\text{minimize}_x \quad \phi_\mu^1(x) \;=\; f(x) + \frac{1}{\mu}\left[\sum_i \left|g_i(x)\right| + \sum_i \left|[h_i(x)]^-\right|\right]$$



minimize $f(x) = \sin(x)$ s.t. $x \geq 0, \quad x \leq 1$

The exact penalty method

Properties of the exact penalty method:

- for sufficiently small penalty parameter, the optimum of the modified problem is the optimum of the original one;
- possibly only one iteration in the penalty parameter needed if size of μ is known in advance;
- this is a non-smooth problem!

This is an efficient method
if (but only if!) a solver for nonsmooth problems is available!

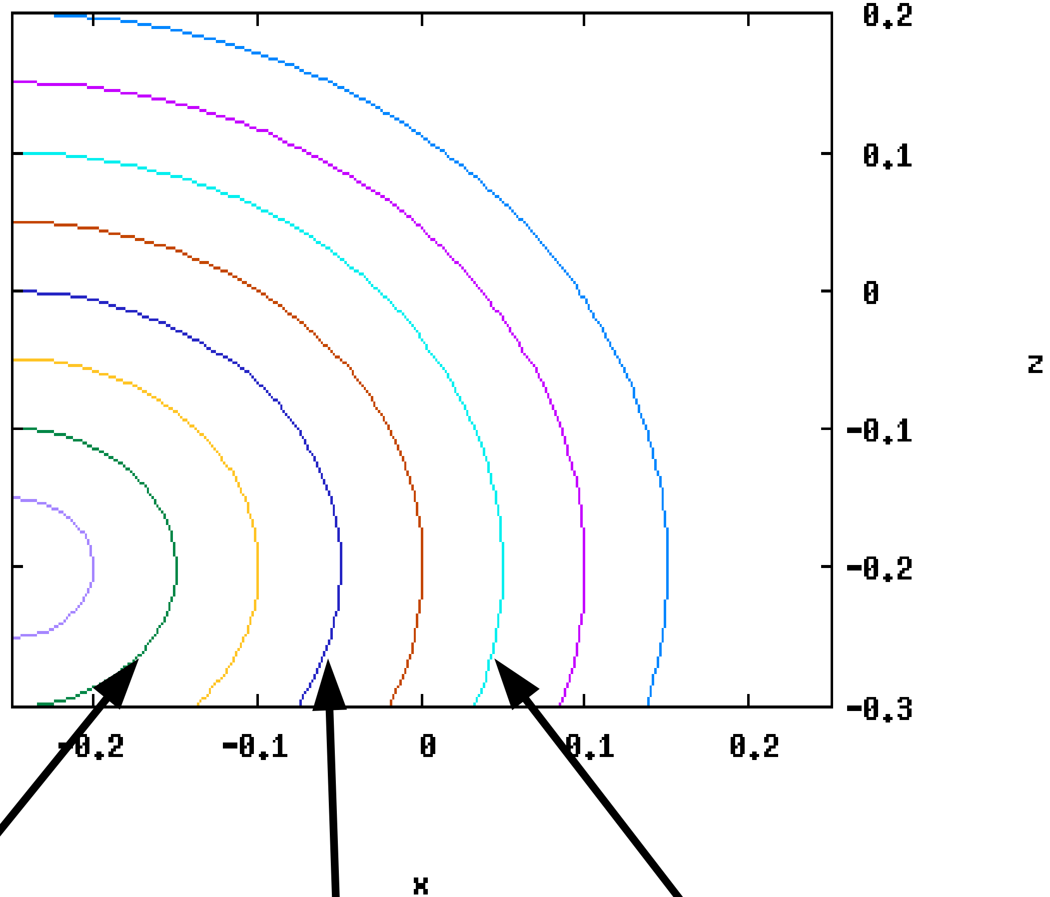
Part 12

Theory of Inequality-Constrained Problems

$$\begin{array}{lllll} \text{minimize} & f(x) & & & \\ & g_i(x) & = & 0, & i = 1, \ldots, n_e \\ & h_i(x) & \geq & 0, & i = 1, \ldots, n_i \end{array}$$

Lagrange multipliers

Consider a (single) constraint *h(x)* as a function everywhere:

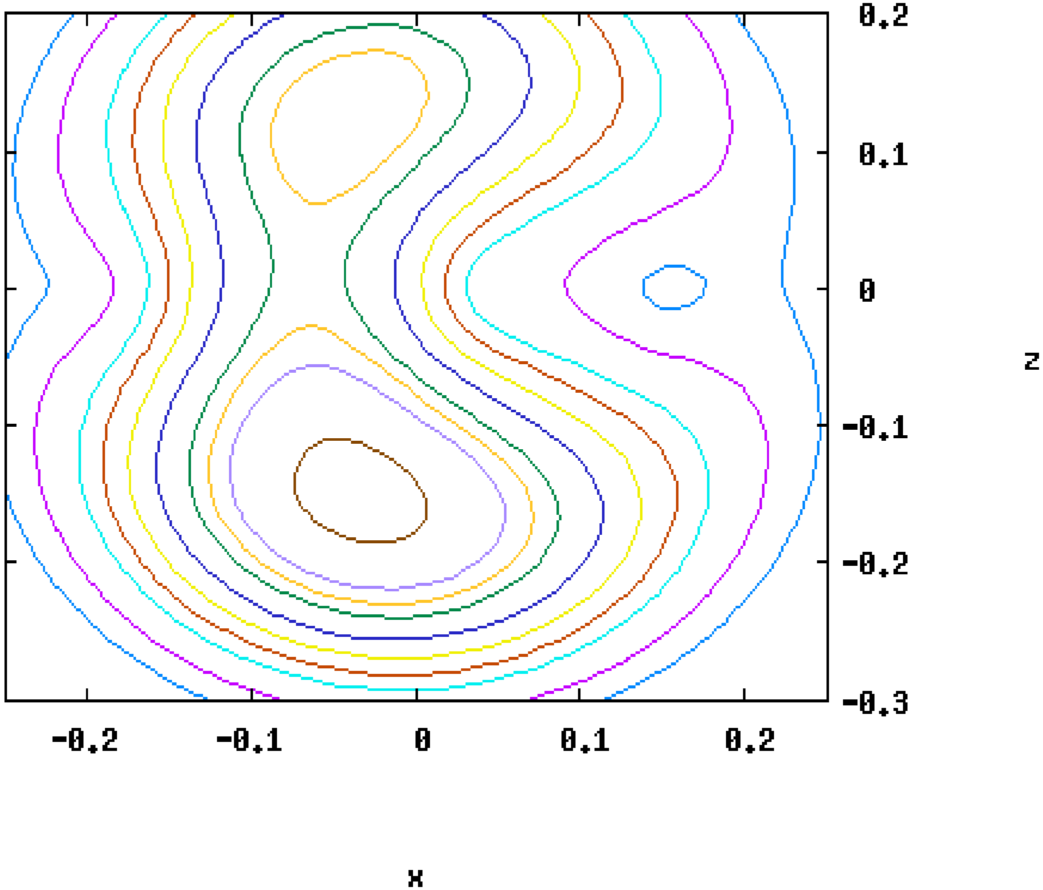


h(x,z)=-0.1   *h(x,z)=0*   *h(x,z)=0.1*

$$h(\vec{x}) = \|\vec{x} - \vec{x}_0\| - L_{\mathrm{rod}} \geq 0$$

Lagrange multipliers

Now look at the objective function *f(x)*:

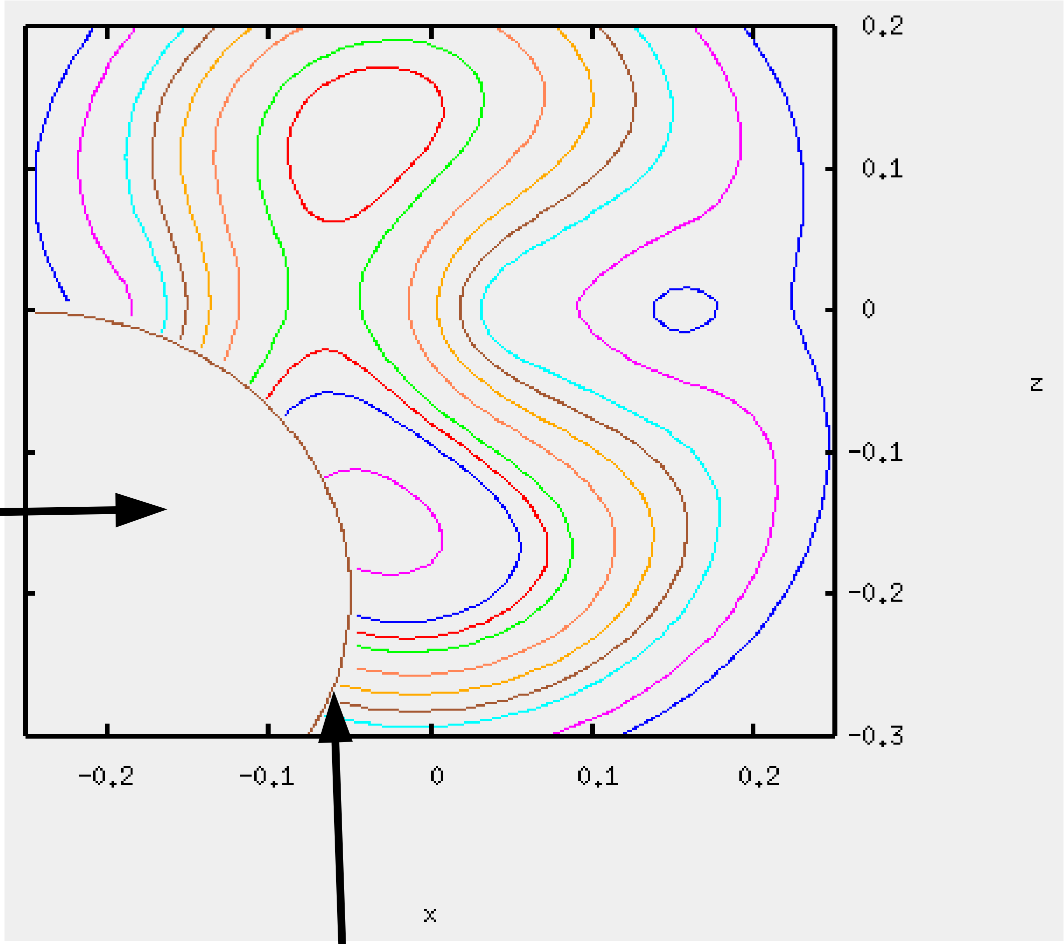


$$f(\vec{x}) = \sum_{i=1}^{3} \frac{1}{2} D \left(\|\vec{x} - \vec{x}_i\| - L_0 \right)^2$$

Lagrange multipliers

Now both *f(x)*, *h(x)* for the case of a rod of minimal length 20cm:

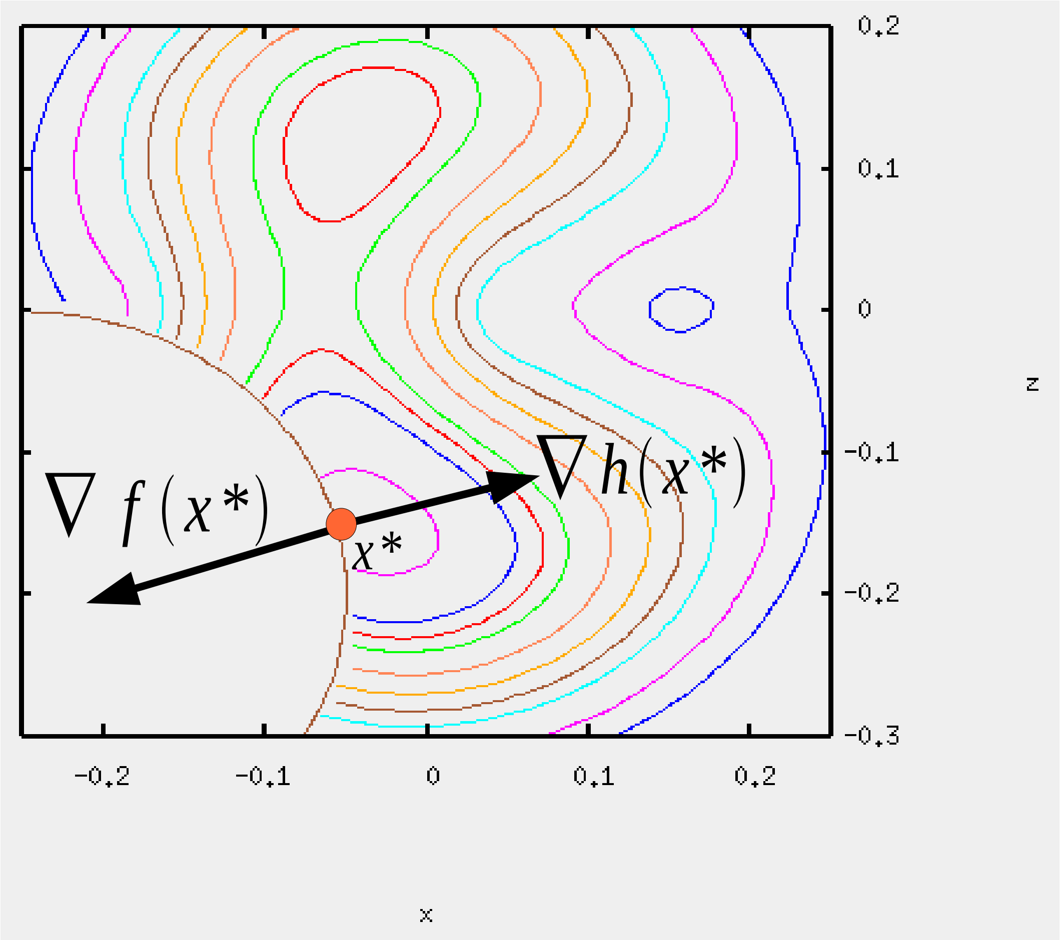
infeasible region



$h(x,z)=0$ with $L_{rod}=20cm$

Lagrange multipliers

Could this be a solution *x**?**



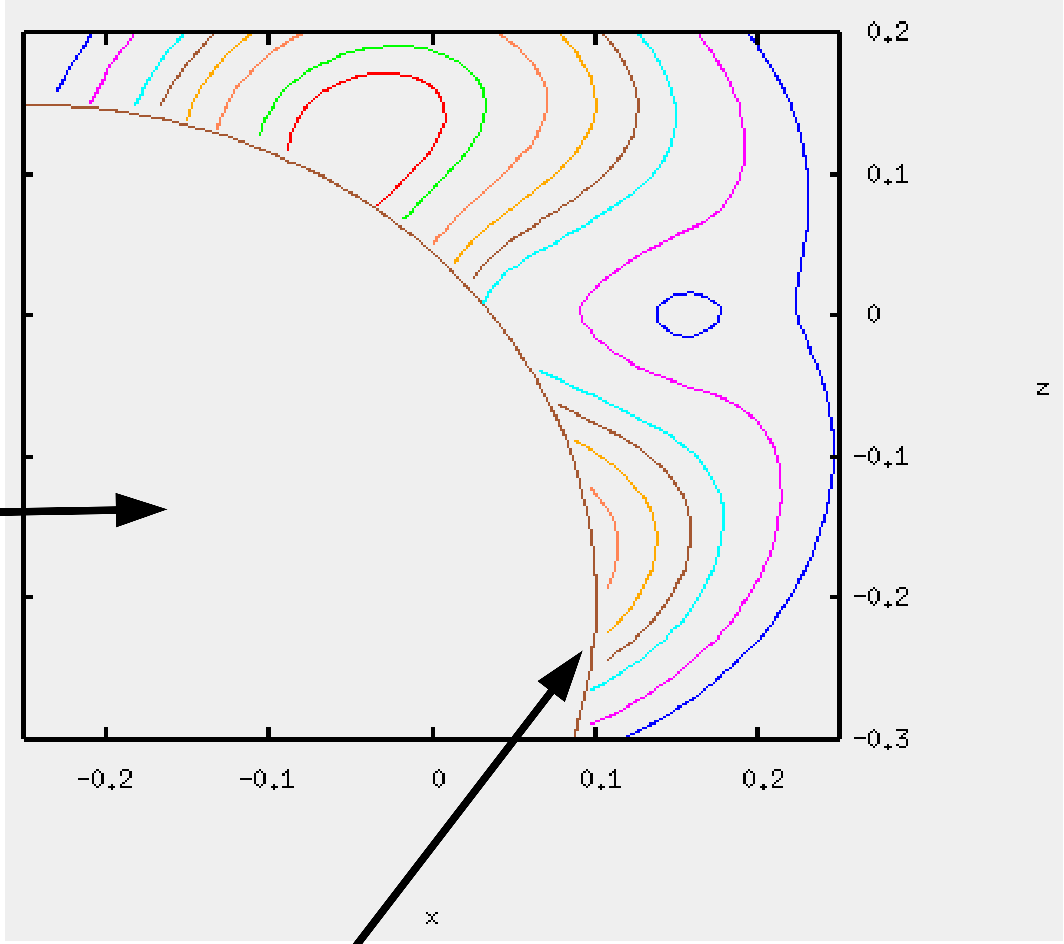
Answer: No – moving into the feasible direction would also reduce *f(x)*.

Rather, the solution will equal the unconstrained one, and the inequality constraint will be inactive at the solution.

Lagrange multipliers

Now both *f(x)*, *h(x)* for the case of a rod of minimal length 35cm:

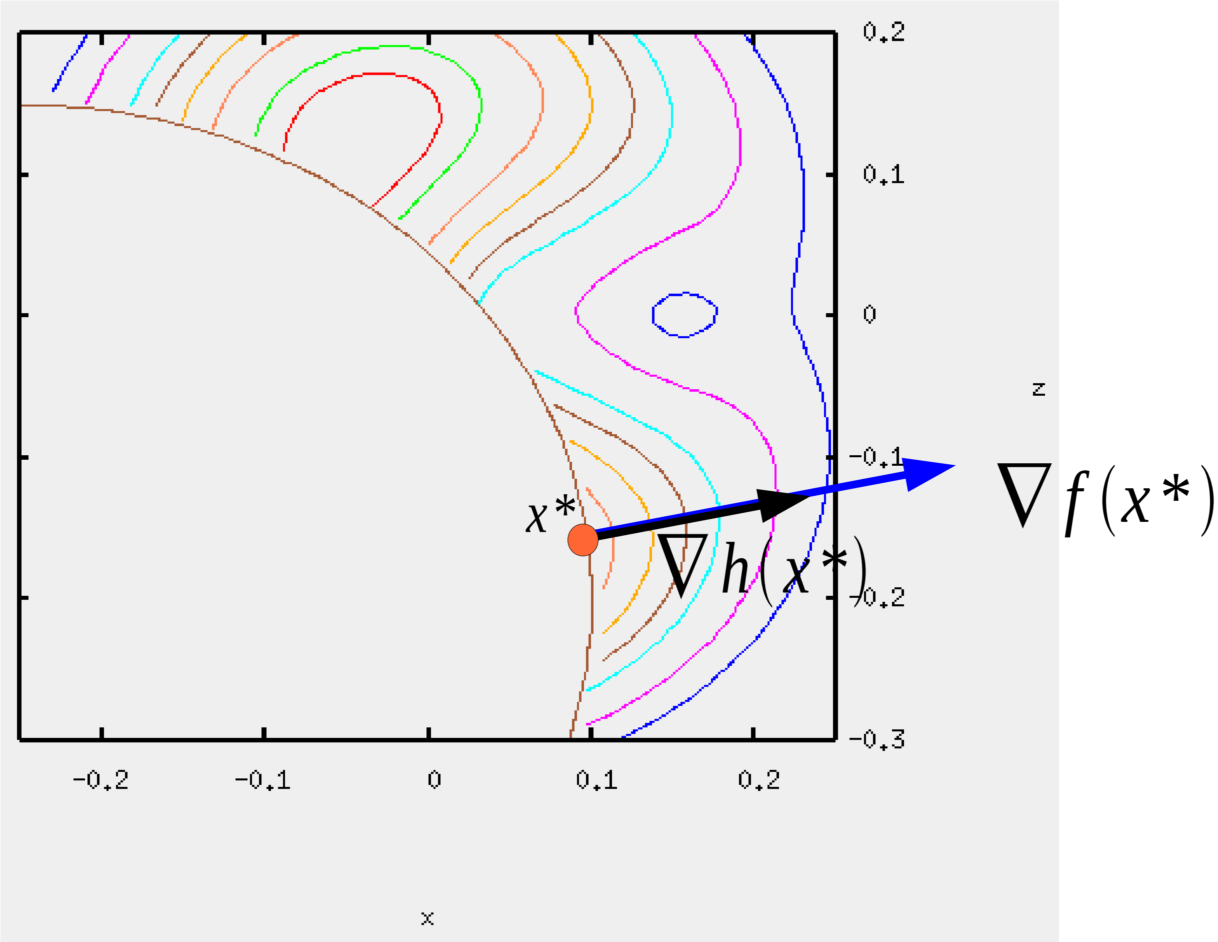
infeasible
region



$h(x,z)=0$ with $L_{rod}=35cm$

Lagrange multipliers

Could this be a solution *x?**



Answer: Yes – moving into the feasible direction would increase *f(x)*.

Note: The gradients of *h* and *f* are parallel and in the same direction.

Lagrange multipliers

Conclusion:

- The solution could be somewhere where the constraint is not active
- If the constraint *is* active at the solution: gradients of f and g are parallel, but **not** antiparallel

In mathematical terms: The (local) solutions of

$$\text{minimize} \quad f(\vec{x}) = E(x,z) = \sum_i E_{\text{spring},i}(x,z) + E_{\text{pot}}(x,z)$$

$$h(\vec{x}) = \|\vec{x} - \vec{x}_0\| - L_{\text{rod}} \geq 0$$

are where one of the following conditions hold for some *λ,μ*:

$$\begin{aligned} \nabla f(x) - \mu \cdot \nabla h(x) &= 0 \\ h(x) &= 0 \\ \mu &\geq 0 \end{aligned} \qquad \text{or} \qquad \begin{aligned} \nabla f(x) &= 0 \\ h(x) &> 0 \end{aligned}$$

Lagrange multipliers

Conclusion, take 2: Solutions are where either

$$\begin{aligned} \nabla f(x) - \mu \cdot \nabla h(x) &= 0 \\ h(x) &= 0 \\ \mu &\geq 0 \end{aligned} \quad \text{or} \quad \begin{aligned} \nabla f(x) &= 0 \\ h(x) &> 0 \end{aligned}$$

which could also be written like so:

$$\begin{aligned} \nabla f(x) - \mu \cdot \nabla h(x) &= 0 \\ h(x) &= 0 \\ \mu &\geq 0 \end{aligned} \quad \text{or} \quad \begin{aligned} \nabla f(x) - \mu \cdot \nabla h(x) &= 0 \\ h(x) &> 0 \\ \mu &= 0 \end{aligned}$$

(constraint is active) (constraint is inactive)

Lagrange multipliers

Conclusion, take 3: Solutions are where

$$\begin{aligned} \nabla f(x) - \mu \cdot \nabla h(x) &= 0 \\ h(x) &= 0 \\ \mu &\geq 0 \end{aligned} \qquad \text{or} \qquad \begin{aligned} \nabla f(x) - \mu \cdot \nabla h(x) &= 0 \\ h(x) &> 0 \\ \mu &= 0 \end{aligned}$$

or written differently:

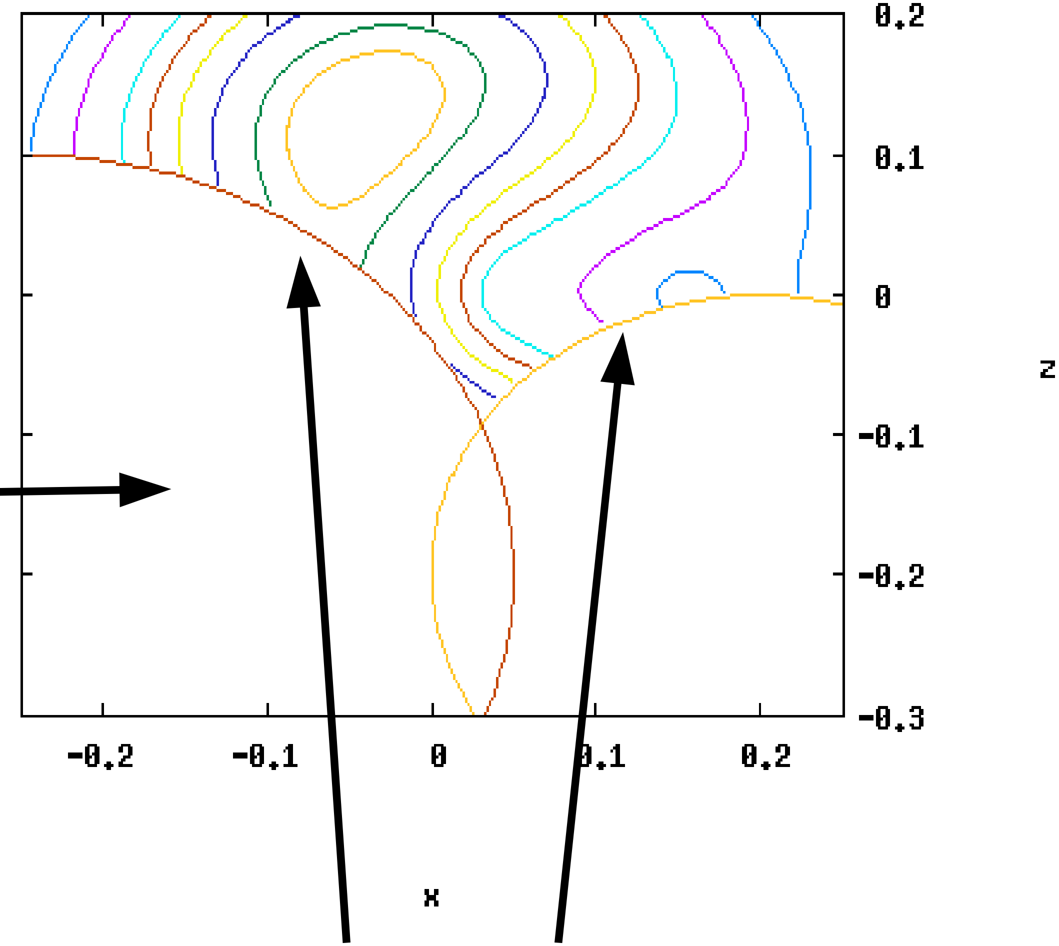
$$\begin{aligned} \nabla f(x) - \mu \cdot \nabla h(x) &= 0 \\ h(x) &\geq 0 \\ \mu &\geq 0 \\ \mu\, h(x) &= 0 \end{aligned}$$

Note: The last condition is called *complementarity*.

Lagrange multipliers

Same idea, but this time with two minimum length elements:

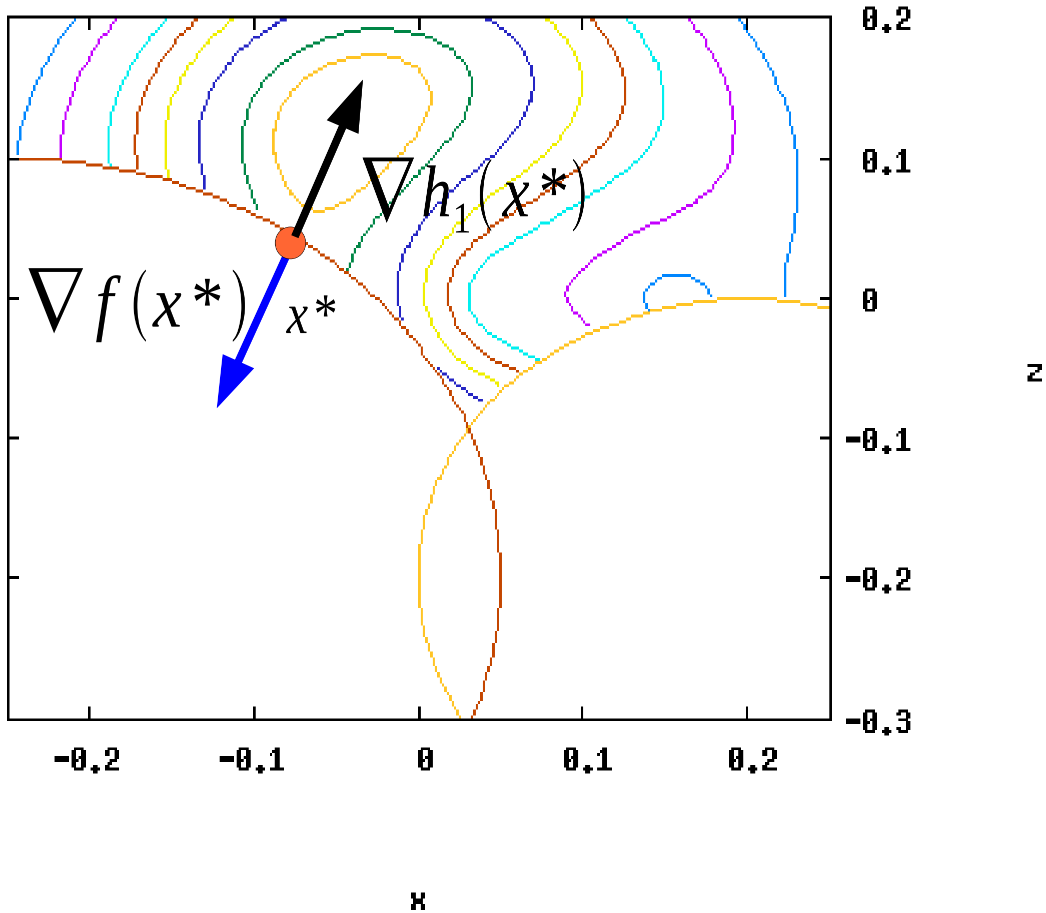
infeasible
region



$h_1(x,z)=0$ $h_2(x,z)=0$

Lagrange multipliers

Could this be a solution *x?**



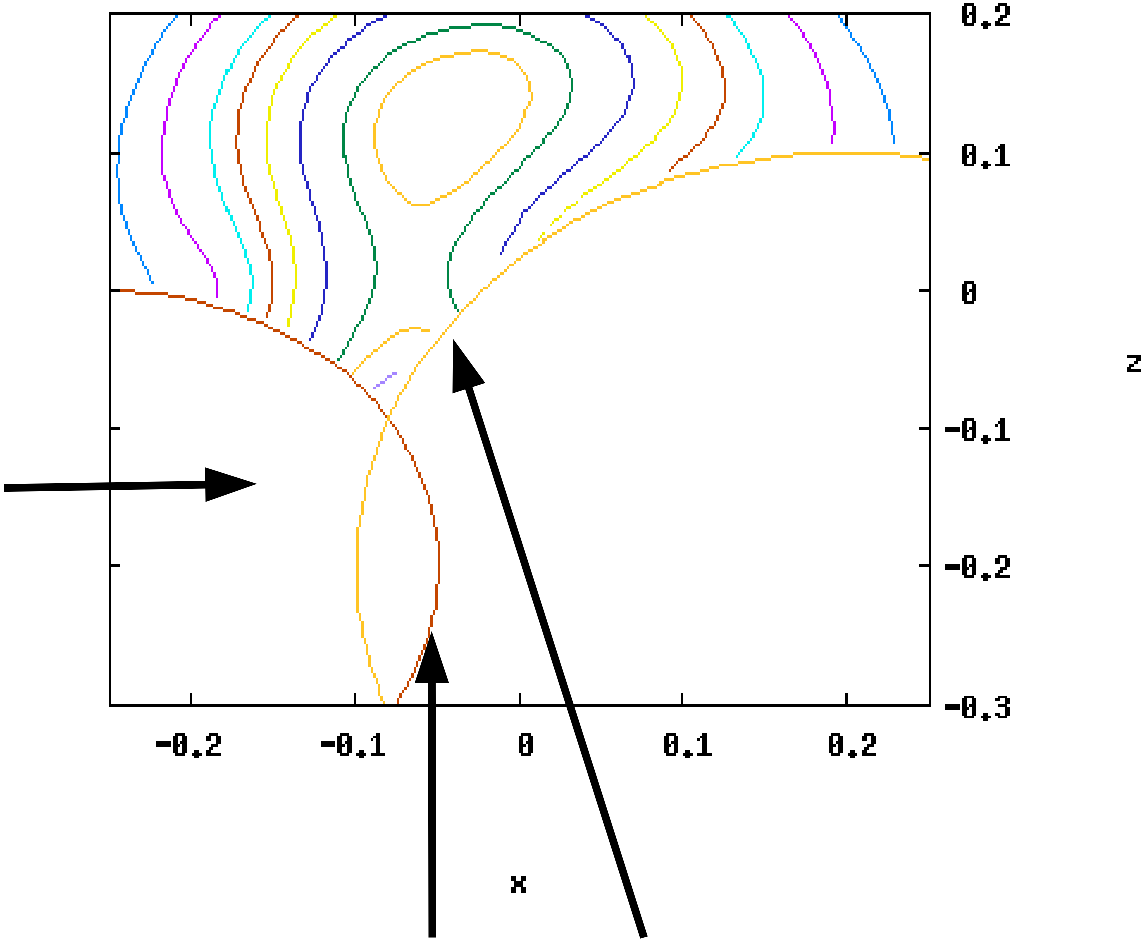
Answer: No – moving into the feasible direction would decrease *f(x)*.

Note: The gradient of *f* is antiparallel to the gradient of h_1. h_2 is an inactive constraint so doesn't matter here.

Lagrange multipliers

Same idea, but this time with two different minimum length elements:

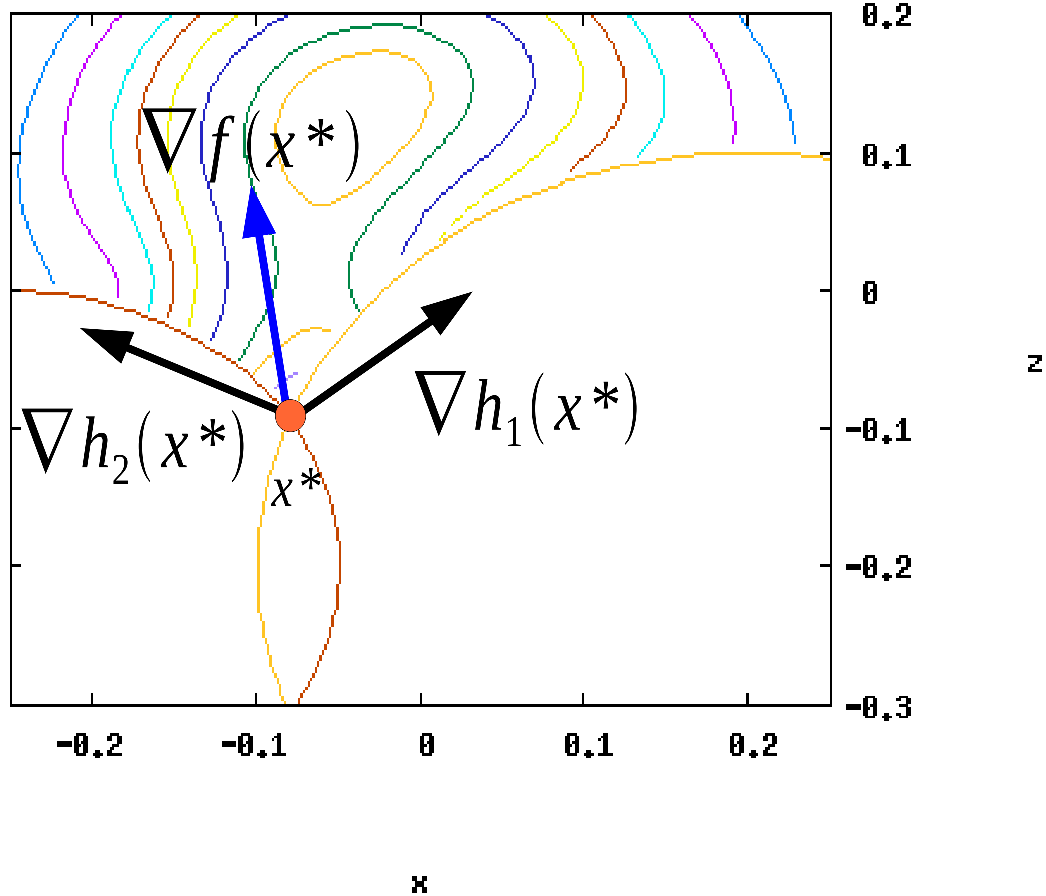
infeasible region



$h_1(x,z)=0$ $\qquad$ $h_2(x,z)=0$

Lagrange multipliers

Could this be a solution *x*?*



Answer: Yes – moving into the feasible direction would increase $f(x)$.

Note: The gradient of f is a linear combination (with positive multiples) of the gradients of h_1 and h_2.

Constraint Qualification: LICQ

Definition:

We say that at a point x the *linear independence constraint qualification* (LICQ) is satisfied if

$$\{\nabla g_i(x)\}_{i=1\ldots n_e}, \{\nabla h_i(x)\}_{i=1\ldots n_i, i \text{ active at } x}$$

is a set of linearly independent vectors.

Note: This is equivalent to saying that the matrix of gradients of all active constraints,

$$A = \begin{bmatrix} [\nabla g_1(x)]^T \\ \vdots \\ [\nabla g_{n_e}(x)]^T \\ [\nabla h_{\text{first active } i}(x)]^T \\ \vdots \\ [\nabla h_{\text{last active } i}(x)]^T \end{bmatrix}$$

has full row rank (i.e. its rank is n_e + # of active ineq. constraints).

First-order necessary conditions

Theorem:

Suppose that x^* is a local solution of

$$\begin{aligned} \text{minimize} \quad & f(x) && f(x): \mathbb{R}^n \to \mathbb{R} \\ & g(x) = 0, && g(x): \mathbb{R}^n \to \mathbb{R}^{n_e} \\ & h(x) \geq 0, && h(x): \mathbb{R}^n \to \mathbb{R}^{n_i} \end{aligned}$$

and suppose that at this point the LICQ holds. Then there exist unique Lagrange multipliers so that the following conditions are satisfied:

$$\begin{aligned} \nabla f(x) - \lambda \cdot \nabla g(x) - \mu \cdot \nabla h(x) &= 0 \\ g(x) &= 0 \\ h(x) &\geq 0 \\ \mu &\geq 0 \\ \mu_i h_i(x) &= 0 \end{aligned}$$

Note: These are often called the *Karush-Kuhn-Tucker (KKT)* conditions.

First-order necessary conditions

Note: By introducing a Lagrangian

$$L(x,\lambda,\mu)=f(x)-\lambda^T g(x)-\mu^T h(x)$$

the first two of the necessary conditions

$$\begin{aligned}
\nabla f(x)-\lambda\cdot\nabla g(x)-\mu\cdot\nabla h(x) &= 0\\
g(x) &= 0\\
h(x) &\geq 0\\
\mu &\geq 0\\
\mu_i h_i(x) &= 0
\end{aligned}$$

follow from requiring that $\nabla_z L(z)$ with $z=\{x,\lambda,\mu\}$, but not the rest.

Consequence: We can not hope to find simple Newton-based methods like SQP to solve inequality-constrained problems.

First-order necessary conditions

Note: The necessary conditions

$$\begin{aligned} \nabla f(x) - \lambda \cdot \nabla g(x) - \mu \cdot \nabla h(x) &= 0 \\ g(x) &= 0 \\ h(x) &\geq 0 \\ \mu &\geq 0 \\ \mu_i h_i(x) &= 0 \end{aligned}$$

imply that there is a unique set of (active) Lagrange multipliers so that

$$\nabla f(x) = A^T \begin{pmatrix} \lambda \\ [\mu]_{\text{active}} \end{pmatrix}$$

where A is the matrix of gradients of active constraints. An alternative way of saying this is

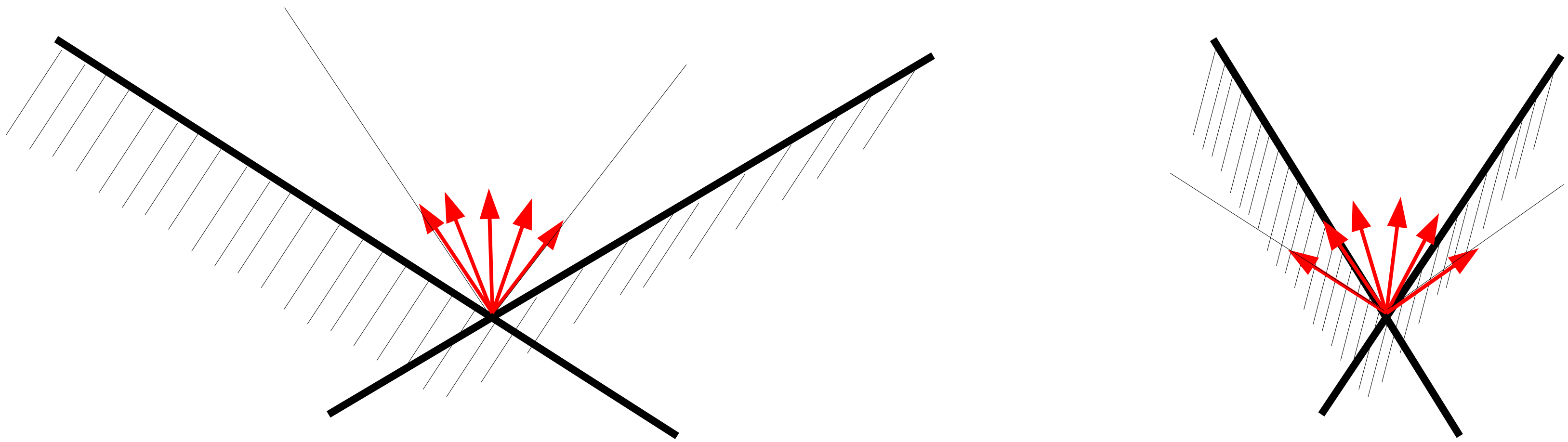
$$\nabla f(x) \in \text{span}\,(\text{rows of}\,(A))$$

 However, the opposite is not true: Multipliers must also satisfy $\mu_i \geq 0$

First-order necessary conditions

A more refined analysis: Consider the constraints

$$h_1(x) = x_2 - a x_1 \geq 0, \qquad h_2(x) = x_2 + a x_1 \geq 0$$



Intuitively (consider the isocontours), the vertex point x^* is optimal if the direction of steepest ascent $\nabla f(x)$ is a member of the family of red vectors above. That is, let F_0 be the *cone*

$$F_0(x^*) = \{ w \in \mathbb{R}^n : \ w = \mu_1 \nabla h_1(x^*) + \mu_2 h_2(x^*), \ \mu_1 \geq 0, \ \mu_2 \geq 0 \}$$

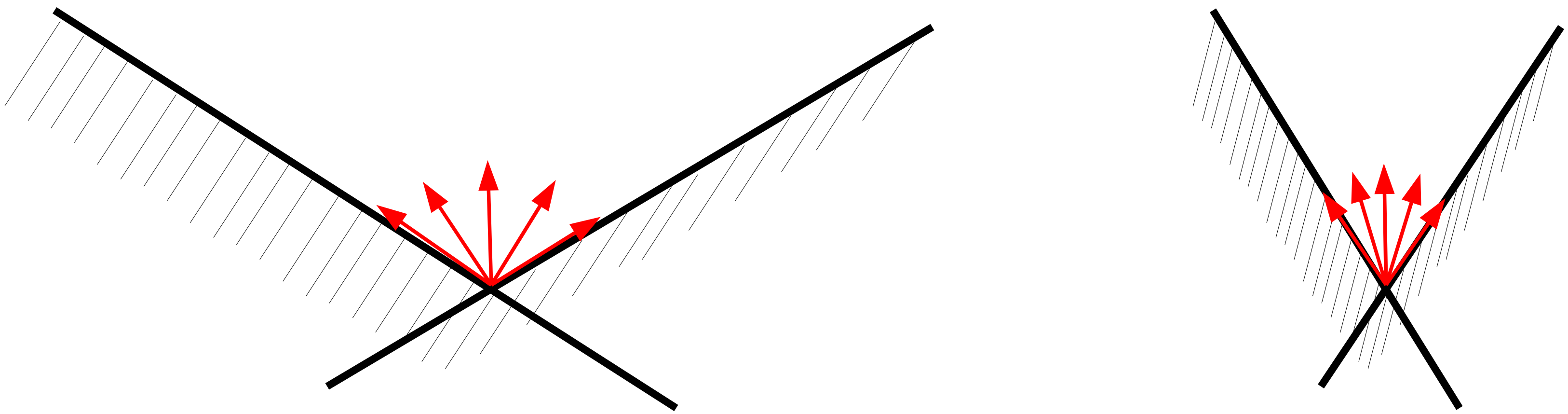
Then x^* is optimal if

$$\nabla f(x^*) \in F_0(x^*)$$

First-order necessary conditions

A more refined analysis: Consider the constraints

$$h_1(x) = x_2 - a x_1 \geq 0, \qquad h_2(x) = x_2 + a x_1 \geq 0$$



Note: We can write things slightly different if we define

$$F_1(x^*) = \{ w \in \mathbb{R}^n : \; w^T a \geq 0 \;\; \forall \, a \in F_0(x^*) \}$$

i.e. the set of vectors that form angles less than 90 degrees with all vectors in F_0. This set can also be written as

$$F_1(x^*) = \{ w \in \mathbb{R}^n : \; w^T \nabla h_1(x^*) \geq 0, \; w^T \nabla h_2(x^*) \geq 0 \}$$

First-order necessary conditions

A more refined analysis: If the problem also has equality constraints

$$g(x) = 0, \qquad h_1(x) \geq 0, \qquad h_2(x) \geq 0$$

all of which are active at x^*, then the cone F_1 is

$$F_1(x^*) = \{ w \in \mathbb{R}^n : \ w^T \nabla g(x^*) = 0, \ w^T \nabla h_1(x^*) \geq 0, \ w^T \nabla h_2(x^*) \geq 0 \}$$

In general:

$$F_1(x^*) = \left\{ \begin{array}{ll} w \in \mathbb{R}^n : & w^T \nabla g_i(x^*) = 0, \quad i = 1, \dots, n_e \\ & w^T \nabla h_i(x^*) \geq 0, \quad i = 1, \dots, n_i, \text{ constraint } i \text{ is active at } x^* \end{array} \right\}$$

First-order necessary conditions

Theorem (a different version of the first order necessary conditions):
If x^* is a local solution and if the LICQ hold at this point, then

$$\nabla f(x^*)^T w \geq 0 \qquad \forall w \in F_1(x^*)$$

In other words: Whatever direction w in F_1 we go into from x^*, the objective function to first order stays constant or increases.

Note: This is a necessary condition, but not sufficient. If $f(x)$ stays constant to first order it may still decrease in higher order Taylor terms to make x^* a local maximum or saddle point. On the other hand, if x^* **is** a solution, then the condition above has to be satisfied.

Second-order necessary conditions

Definition:

Let x^* be a local solution of an inequality constrained problem satisfying

$$
\begin{aligned}
\nabla f(x) - \lambda \cdot \nabla g(x) - \mu \cdot \nabla g(x) &= 0 \\
g_i(x) &= 0, \quad i = 1 \ldots n_e \\
h_i(x) &\geq 0, \quad i = 1 \ldots n_i \\
\mu_i &\geq 0, \quad i = 1 \ldots n_i \\
\mu_i h_i(x) &= 0, \quad i = 1 \ldots n_i
\end{aligned}
$$

We say that *strict complementarity* holds if for each inequality constraint i *exactly one* of the following conditions is true:

- $\mu_i = 0$
- $h_i(x^*) = 0$

In other words, we require that the Lagrange multiplier is nonzero for all active inequality constraints.

Second-order necessary conditions

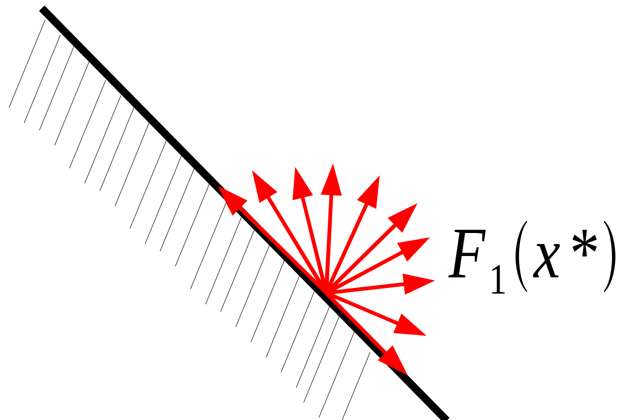
Definition:

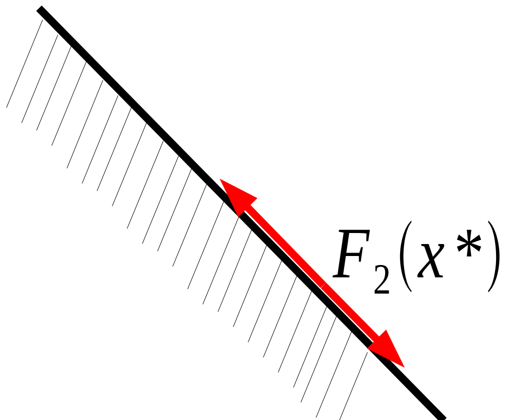
Let x^* be a local solution and assume that strict complementarity holds. Then define as before

$$F_1(x^*) = \left\{ \begin{array}{ll} w \in \mathbb{R}^n : & w^T \nabla g_i(x^*) = 0, \quad i = 1, \dots, n_e \\ & w^T \nabla h_i(x^*) \geq 0, \quad i = 1, \dots, n_i, \text{ constraint } i \text{ is active at } x^* \end{array} \right\}$$

and the subspace of all *tangential* directions as

$$F_2(x^*) = \left\{ \begin{array}{ll} w \in \mathbb{R}^n : & w^T \nabla g_i(x^*) = 0, \quad i = 1, \dots, n_e \\ & w^T \nabla h_i(x^*) = 0, \quad i = 1, \dots, n_i, \text{ constraint } i \text{ is active at } x^* \end{array} \right\}$$





Second-order necessary conditions

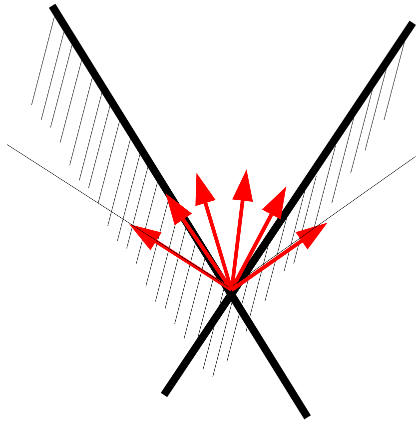
Note:

The subspace of all *tangential* directions

$$F_2(x^*) = \left\{ w \in \mathbb{R}^n : \begin{array}{l} w^T \nabla g_i(x^*) = 0, \quad i = 1,\dots,n_e \\ w^T \nabla h_i(x^*) = 0, \quad i = 1,\dots,n_i, \text{ constraint } i \text{ is active at } x^* \end{array} \right\}$$

can be empty (i.e. contain only the zero vector) if n or more constraints are active at x^*.

Example:



Here, F_1 is a nonempty set, but F_2 contains only the zero vector.

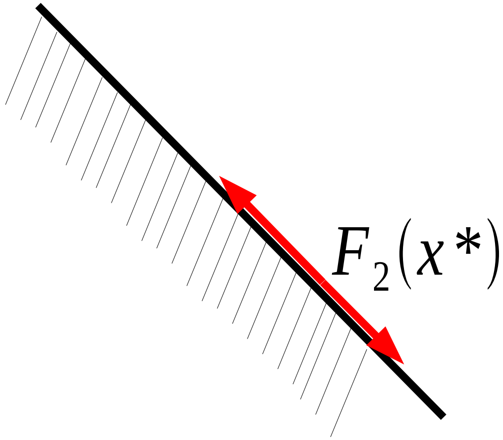
Second-order necessary conditions

Theorem (necessary conditions):

Let *x** be a local solution that satisfies the first order necessary conditions with unique Lagrange multipliers. Assume that strict complementarity holds. Then

$$w^T \nabla_x^2 L(x^*, \lambda^*, \mu^*) w =$$
$$= w^T \left[\nabla_x^2 f(x^*) - \lambda^{*T} \nabla_x^2 g(x^*) - \mu^* \nabla_x^2 h(x^*) \right] w \ \geq \ 0$$
$$\forall\, w \in F_2(x^*)$$

Note: This means that *f(x)* can not "curve down" to second order along tangential directions. The first order Conditions imply that it doesn't "slope" in these directions.



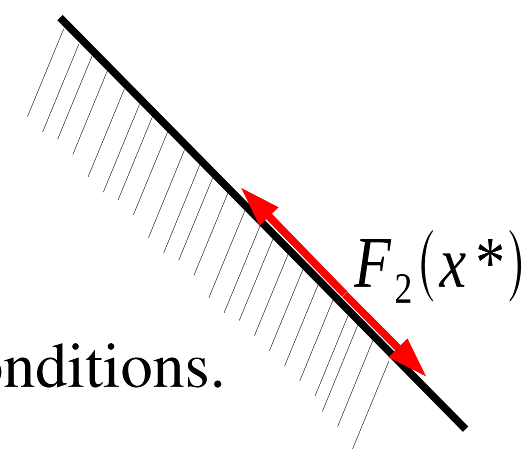
Second-order sufficient conditions

Theorem (sufficient conditions):

Let x^* be a local solution that satisfies the first order necessary conditions with unique Lagrange multipliers. Assume that strict complementarity holds. Then

$$w^T \nabla_x^2 L(x^*, \lambda^*, \mu^*) w =$$
$$= w^T \left[\nabla_x^2 f(x^*) - \lambda^{*T} \nabla_x^2 g(x^*) - \mu^* \nabla_x^2 h(x^*) \right] w \;>\; 0$$
$$\forall\, w \in F_2(x^*), w \neq 0$$

Note: This means that *f(x)* actually "curves up" in a neighborhood of *x**, at least in tangential directions!



For all other directions, we know that *f(x)* slopes up from the first order necessary conditions.

Second-order sufficient conditions

Remark:

If strict complementarity holds, then the definition

$$F_2(x^*) = \left\{ \begin{array}{ll} w \in \mathbb{R}^n : & w^T \nabla g_i(x^*) = 0, \quad i = 1,\ldots,n_e \\ & w^T \nabla h_i(x^*) = 0, \quad i = 1,\ldots,n_i, \text{ constraint } i \text{ is active at } x^* \end{array} \right\}$$

is equivalent to

$$F_2(x^*) = \operatorname{null} A(x^*)$$

with the matrix of gradients of active constraints A. In that case, the second order necessary and sufficient conditions can also be written as

$$\begin{array}{l} Z^T \nabla_x^2 L(x^*, \lambda^*, \mu^*) Z \ \text{ is positive semidefinite} \\ Z^T \nabla_x^2 L(x^*, \lambda^*, \mu^*) Z \ \text{ is positive definite} \end{array}$$

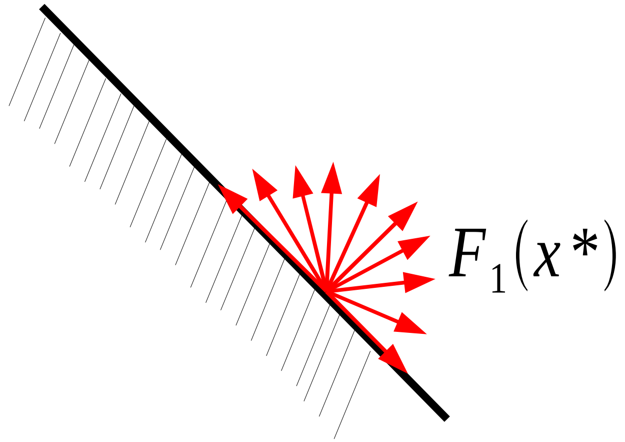
Respectively, where the columns of Z are a basis of the null space of A.

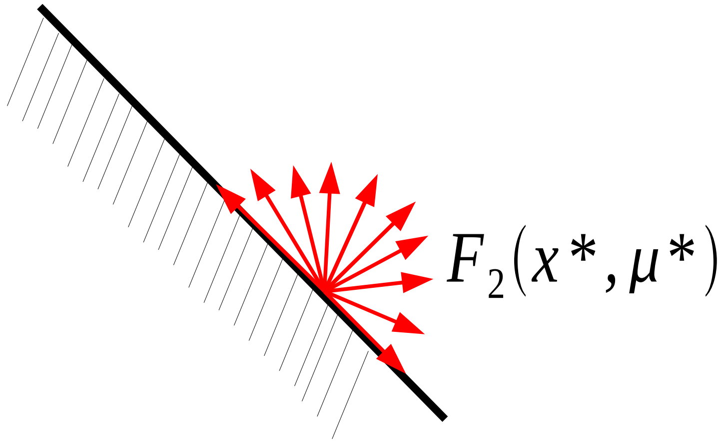
Second-order necessary conditions

Definition (if strict complementarity does not hold):

Let x^* be a local solution at which the KKT conditions with unique Lagrange multiplier hold. Then define

$$F_2(x^*,\mu^*) = \left\{ \begin{array}{l} w \in \mathbb{R}^n \ : \ w^T \nabla g_i(x^*) = 0, \quad i = 1,\ldots,n_e \\ \qquad\qquad w^T \nabla h_i(x^*) = 0, \quad i = 1,\ldots,n_i, \text{ constraint } i \text{ active and } \mu_i^* > 0 \\ \qquad\qquad w^T \nabla h_i(x^*) \geq 0, \quad i = 1,\ldots,n_i, \text{ constraint } i \text{ active and } \mu_i^* = 0 \end{array} \right\}$$





Second-order sufficient conditions

Theorem (sufficient conditions without strict complementarity):

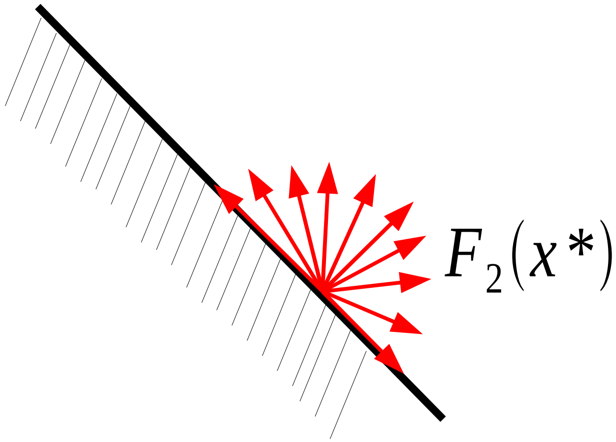
Let *x** be a local solution that satisfies the first order necessary conditions with unique Lagrange multipliers. Assume that strict complementarity *does not* hold. Then

$$w^T \nabla_x^2 L(x^*, \lambda^*, \mu^*) w =$$

$$= w^T \left[\nabla_x^2 f(x^*) - \lambda^{*T} \nabla_x^2 g(x^*) - \mu^* \nabla_x^2 h(x^*) \right] w \;>\; 0$$

$$\forall\, w \in F_2(x^*)$$

Note: This now means that *f(x)* actually "curves up" in a neighborhood of *x**, at least in tangential directions plus all those directions for which we can't infer anything from the first order conditions!



Part 12

Active Set Methods for Convex Quadratic Programs

$$\text{minimize} \quad f(x) = \frac{1}{2} x^T G x + x^T d + e$$

$$\begin{aligned} g_i(x) = a_i^T x - b_i &= 0, \qquad i = 1, \ldots, n_e \\ h_i(x) = \alpha_i^T x - \beta_i &\geq 0, \qquad i = 1, \ldots, n_i \end{aligned}$$

General idea

Note:

Recall that if W^* is the set of active constraints at the solution x^* then the solution of

$$\begin{aligned} \text{minimize} \quad & f(x) = \frac{1}{2} x^T G x + x^T d + e \\ & g_i(x) = a_i^T x - b_i \;=\; 0, \qquad i = 1, \ldots, n_e \\ & h_i(x) = \alpha_i^T x - \beta_i \;\geq\; 0, \qquad i = 1, \ldots, n_i \end{aligned}$$

equals the solution of the following QP:

$$\begin{aligned} \text{minimize} \quad & f(x) = \frac{1}{2} x^T G x + x^T d + e \\ & g_i(x) = a_i^T x - b_i \;=\; 0, \qquad i = 1, \ldots, n_e \\ & h_i(x) = \alpha_i^T x - \beta_i \;=\; 0, \qquad i = 1, \ldots, n_i, i \in W^* \end{aligned}$$

General idea

Definition: Let

$$A = \begin{pmatrix} a_1^T \\ \vdots \\ a_{n_e}^T \\ \alpha_1^T \\ \vdots \\ \alpha_{n_i}^T \end{pmatrix} \qquad B = \begin{pmatrix} b_1 \\ \vdots \\ b_{n_e} \\ \beta_1 \\ \vdots \\ \beta_{n_i} \end{pmatrix} \qquad A|_W = \begin{pmatrix} a_1^T \\ \vdots \\ a_{n_e}^T \\ \alpha_{\text{first inequality in } W}^T \\ \vdots \\ \alpha_{\text{last inequality in } W}^T \end{pmatrix} \qquad B|_W = \begin{pmatrix} b_1 \\ \vdots \\ b_{n_e} \\ \beta_{\text{first inequality in } W} \\ \vdots \\ \beta_{\text{last inequality in } W} \end{pmatrix}$$

then the solution of the inequality-constrained QP equals the solution of the following QP:

$$\text{minimize} \quad f(x) = \frac{1}{2} x^T G x + x^T d + e$$

$$A|_{W*} x - B|_{W*} = 0$$

General idea

Consequence: If we knew the active set W^* at the solution, we could just solve the linearly constrained QP

$$\text{minimize} \quad f(x) = \frac{1}{2} x^T G x + x^T d + e$$
$$A|_{W^*} x - B|_{W^*} = 0$$

and be done in one step.

Problem: Knowing the exact active set W^* requires knowing the solution x^* because W^* is the set of all equality constraints plus those constraints for which

$$h_i(x^*) = 0$$

Solution: Solve a sequence of QPs using working sets W_k that we iteratively refine until we have the exact active set W^*.

The active set algorithm

Algorithm:

- Choose an initial working set W_0 and a point x_0 that is feasible with respect to these constraints
- For k=0, 1, 2, ….:

  - Find the SQP search direction p_k from x_k to the solution x_{k+1} of

$$\text{minimize} \quad f(x) = \frac{1}{2} x^T G x + x^T d + e$$

$$A|_{W_k} x - B|_{W_k} = 0$$

  - If $p_k = 0$ and all $\mu_i \geq 0$ for constraints in W_k then stop
  - Else if $p_k = 0$ but there are $\mu_i < 0$, then drop the inequality with the most negative μ_i from W_k to obtain W_{k+1}
  - Else if $x_k + p_k$ is feasible with respect to W_k then set $x_{k+1} = x_k + p_k$
  - Otherwise, set $x_{k+1} = x_k + \alpha_k p_k$ with

$$\alpha_k = \min \left\{ 1, \min_{i \notin W_k, \alpha_i^T p_k < 0} \frac{\beta_i - \alpha_i^T x_k}{\alpha_i^T p_k} \right\}$$

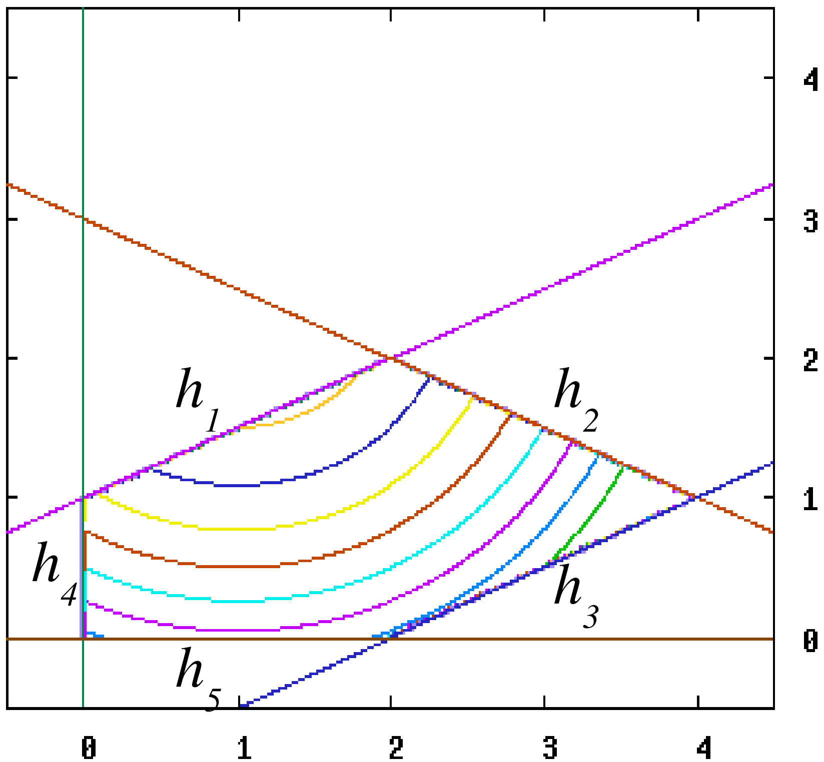
and add the most blocking constraint to the active set W_{k+1}

The active set algorithm

Example:

$$\text{minimize} \quad f(x) = (x_1 - 1)^2 + (x_2 - 2.5)^2$$

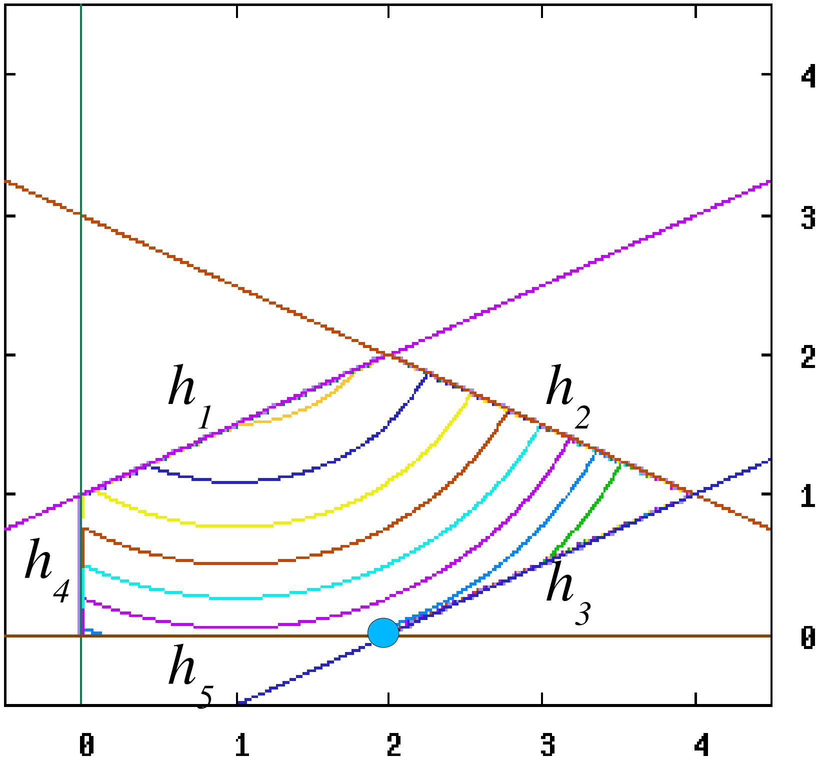
$$\begin{pmatrix} 1 & -2 \\ -1 & -2 \\ -1 & 2 \\ 1 & 0 \\ 0 & 1 \end{pmatrix} x - \begin{pmatrix} -2 \\ -6 \\ -2 \\ 0 \\ 0 \end{pmatrix} \geq 0$$



Choose as initial working set $W_0=\{3,5\}$ and as starting point $x_0=(2,0)^T$.

The active set algorithm

Example: Step 0



$W_0=\{3,5\}$, $x_0=(2,0)^T$.

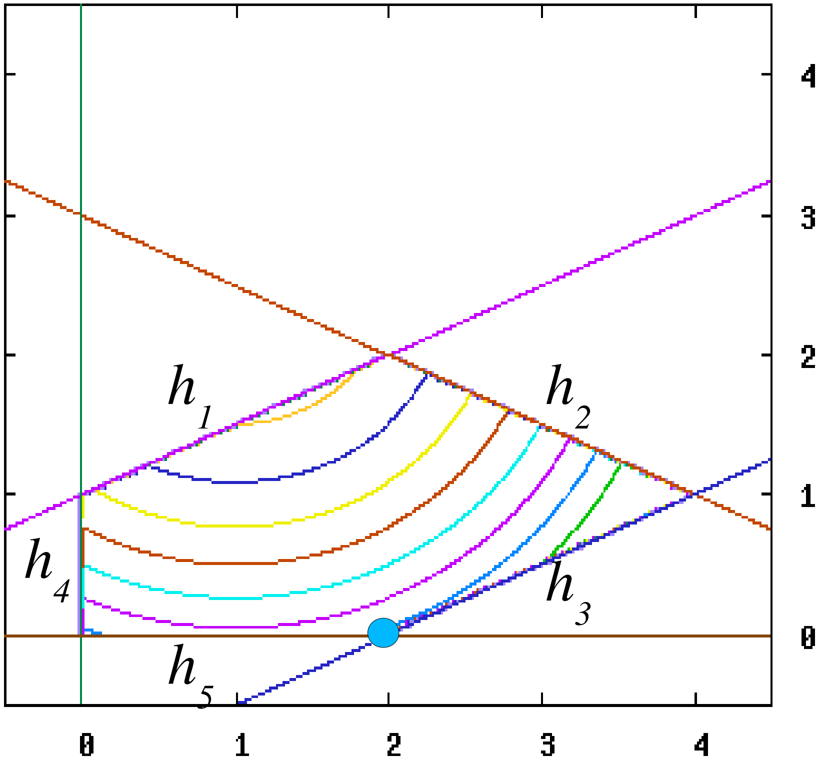
Then: $p_0=(0,0)^T$ because there is no other point that is feasible for W_0

$$\nabla f(x_0)-\mu|_{W_0}^T A|_{W_0}=\begin{pmatrix}2\\-5\end{pmatrix}-\begin{pmatrix}\mu_3\\\mu_5\end{pmatrix}^T\begin{pmatrix}-1 & 2\\0 & 1\end{pmatrix}=0 \quad \text{implies} \quad \begin{pmatrix}\mu_3\\\mu_5\end{pmatrix}=\begin{pmatrix}-2\\-1\end{pmatrix}$$

Consequently: $W_1=\{5\}$, $x_1=(2,0)^T$.

The active set algorithm

Example: Step 1



$W_1=\{5\}$, $x_1=(2,0)^T$.

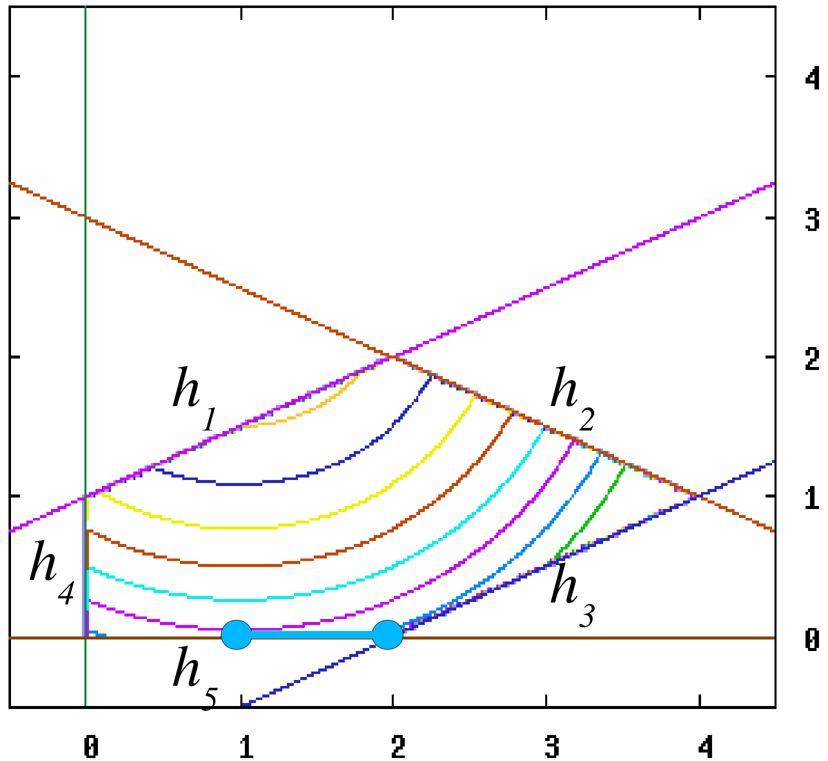
Then: $p_1=(-1,0)^T$ leads to the minimum along the only active constraint.

There are no blocking constraints to get to the point $x_{k+1}=x_k+p_k$

Consequently: $W_2=\{5\}$, $x_2=(1,0)^T$.

The active set algorithm

Example: Step 2



$W_2=\{5\}$, $x_2=(1,0)^T$.

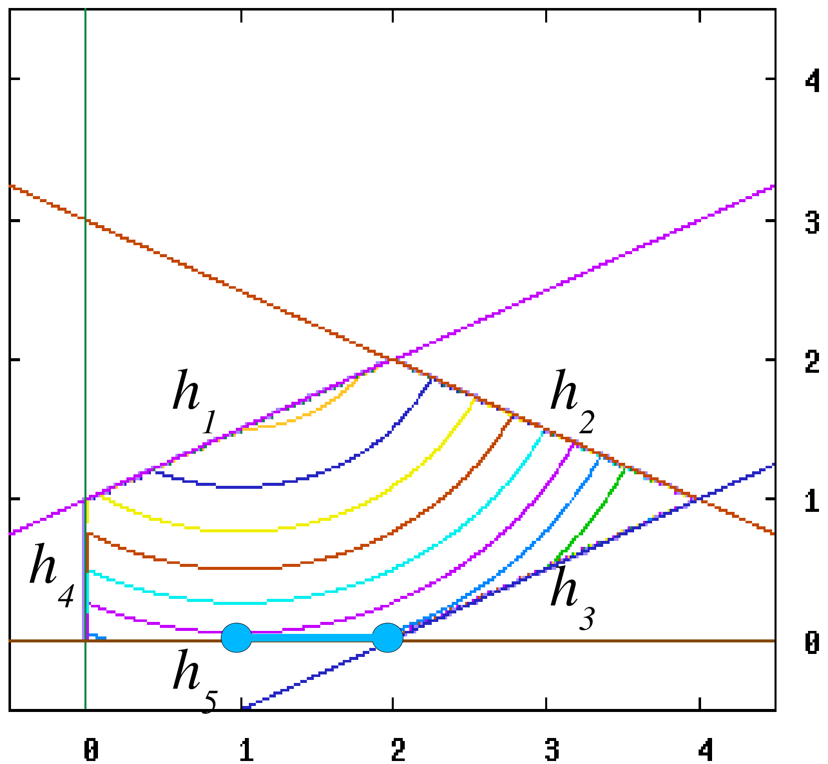
Then: $p_2=(0,0)^T$ because we are at the minimum of the active constraints.

$$\nabla f(x_2)-\mu|_{W_2}^T A|_{W_2}=\begin{pmatrix}0\\-5\end{pmatrix}-(\mu_5)^T\begin{pmatrix}0 & 1\end{pmatrix}=0 \qquad \text{implies} \qquad (\mu_5)=(-5)$$

Consequently: $W_3=\{\}$, $x_3=(1,0)^T$.

The active set algorithm

Example: Step 3



$W_3=\{\}$, $x_3=(1,0)^T$.

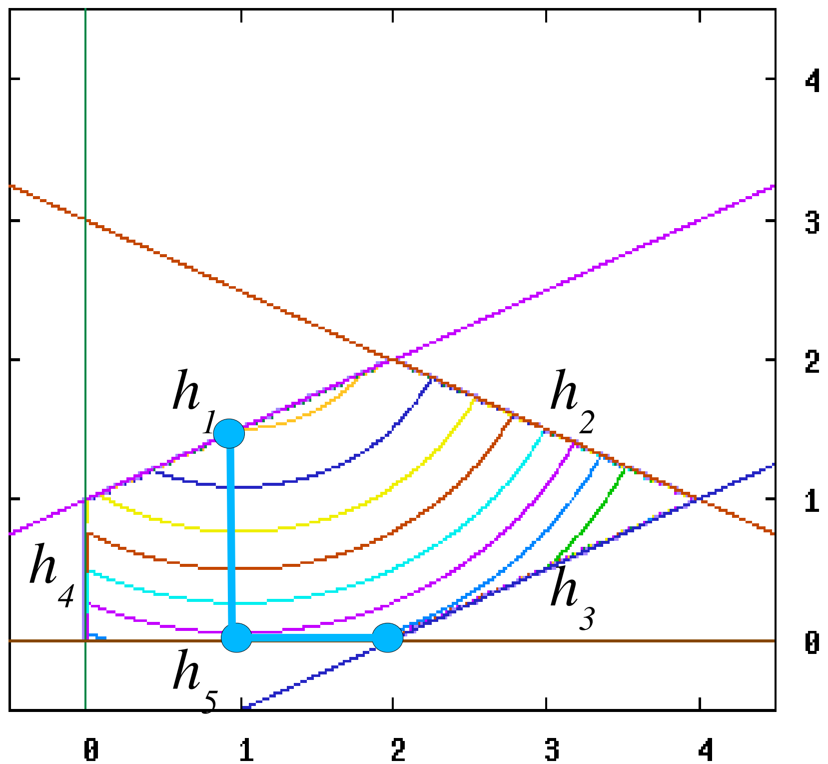
Then: $p_3=(0,2.5)^T$ but this leads out of the feasible region. The first blocking constraint is inequality 1, and the maximal step length is

$$\alpha_3 = 0.6$$

Consequently: $W_4=\{1\}$, $x_4=(1,1.5)^T$.

The active set algorithm

Example: Step 4



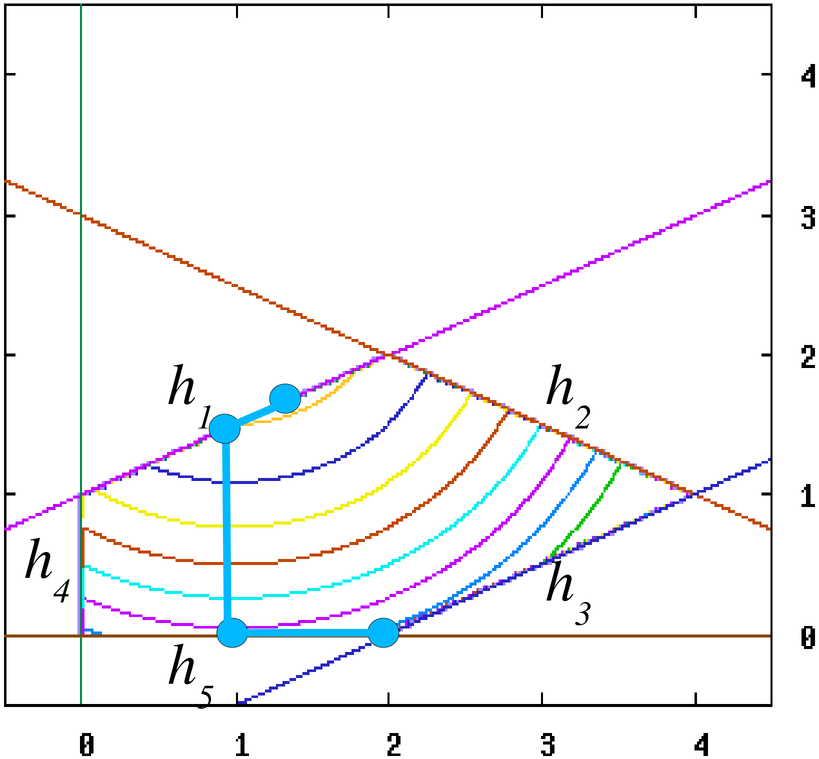
$W_4=\{1\}$, $x_4=(1,1.5)^T$.

Then: $p_4=(0.4,0.2)^T$ is the minimizer along the sole constraint. There are no blocking constraints to get there.

Consequently: $W_5=\{1\}$, $x_5=(1.4,1.7)^T$.

The active set algorithm

Example: Step 5



$W_5 = \{1\}$, $x_5 = (1.4, 1.7)^T$.

Then: $p_5 = (0,0)^T$ because we are already on the minimizer on the constraint. Furthermore,

$$\nabla f(x_5) - \mu|_{W_5}^T A|_{W_5} = \begin{pmatrix} 0.8 \\ -1.6 \end{pmatrix} - \left(\mu_1\right)^T \begin{pmatrix} 1 & -2 \end{pmatrix} = 0 \quad \text{implies} \quad \left(\mu_1\right) = \left(0.8\right) \geq 0$$

Consequently: This is the solution.

The active set algorithm

Theorem:

If G is strictly positive definite (i.e. the objective function is strictly convex), then $W_k \neq W_l$ for $k \neq l$.

Consequently (because there are only finitely many possible working sets), the active set algorithm terminates in a finite number of steps.

Note:

In practice it may be that G is indefinite, and that for some iterations the matrix $Z_k^T G Z_k$ is indefinite as well. We know that at the solution, $Z_*^T G Z_*$ is positive semidefinite, however. In that case, we can't guarantee termination or convergence.

There are, however, Hessian modification techniques to deal with this situation.

The active set algorithm

Remark:

In the active set method, we only change the working set W_k by at most one element in each iteration.

One may be tempted to remove *all* constraints with negative Lagrange multipliers at once, or add several constraints at the same time when they become active.

However, in cases like this it can't be guaranteed any more that $W_k \neq W_l$ for $k \neq l$ and *cycling* may happen, i.e. we cycle between the same points and sets x_k, W_k.

Active set SQP methods for general nonlinear problems

For equality constrained problems of the form

$$\begin{array}{lll} \text{minimize} & f(x) & f(x):\mathbb{R}^n \to \mathbb{R} \\ & g(x) = 0, & g(x):\mathbb{R}^n \to \mathbb{R}^{n_e} \end{array}$$

the SQP method could be employed. It repeatedly solves linear-quadratic problems of the form

$$\min_x \; m_k(p_k^x) = L(x_k,\lambda_k) + \nabla_x L(x_k,\lambda_k)^T p_k^x + \frac{1}{2} p_k^{xT} \nabla_x^2 L(x_k,\lambda_k)\, p_k^x$$

$$g(x_k) + \nabla g(x_k)^T p_k^x = 0$$

Here, each subproblem (a single SQP step) could be solved in one iteration by solving a saddle point linear system.

Active set SQP methods for general nonlinear problems

For inequality constrained problems of the form

$$\begin{aligned} \text{minimize} \quad & f(x) \\ & g_i(x) = 0, \qquad i = 1,\ldots,n_e \\ & h_i(x) \geq 0, \qquad i = 1,\ldots,n_i \end{aligned}$$

we repeatedly solve linear-quadratic problems of the form

$$\begin{aligned} \min_x \quad & m_k(p_k^x) = L(x_k,\lambda_k) + \nabla_x L(x_k,\lambda_k)^T p_k^x + \frac{1}{2} p_k^{xT} \nabla_x^2 L(x_k,\lambda_k) p_k^x \\ & g(x_k) + \nabla g(x_k)^T p_k^x = 0 \\ & h(x_k) + \nabla h(x_k)^T p_k^x \geq 0 \end{aligned}$$

Each of these inequality constrained quadratic problems can be solved using the active set method, and after we have the exact solution of this approximate problem we can re-linearize around this point for the next sub-problem.

Active set SQP methods for general nonlinear problems

Note: Each time we solve a problem like

$$\min_x \; m_k(p_k^x) = L(x_k,\lambda_k) + \nabla_x L(x_k,\lambda_k)^T p_k^x + \frac{1}{2} p_k^{xT} \nabla_x^2 L(x_k,\lambda_k) p_k^x$$

$$g(x_k) + \nabla g(x_k)^T p_k^x = 0$$

$$h(x_k) + \nabla h(x_k)^T p_k^x \geq 0$$

we have to do several active set iterations, though we can start with the previous step's final working set and solution point.

Nevertheless, this is not going to be cheap, though it is comparable to iterating over penalty/barrier parameters.

Parts 11-12

Summary of methods for inequality-constrained Problems

$$
\begin{aligned}
\text{minimize} \quad & f(x) \\
& g_i(x) = 0, \qquad i = 1,\ldots,n_e \\
& h_i(x) \geq 0, \qquad i = 1,\ldots,n_i
\end{aligned}
$$

Summary of methods

There are two general methods for inequality-constrained problems:

- Penalty/barrier methods (e.g. the quadratic penalty and logarithmic barrier methods) convert the constrained problem into an unconstrained one that can be solved with the techniques we already know.

  Barrier methods are able to ensure that intermediate iterates remain feasible with respect to inequality constraints

- Lagrange multiplier formulations give rise to active set methods

- Both kinds of methods are expensive. Penalty/barrier methods are simpler to implement but can only find minima located at the boundary of the feasible set at the price of dealing with ill-conditioned problems.

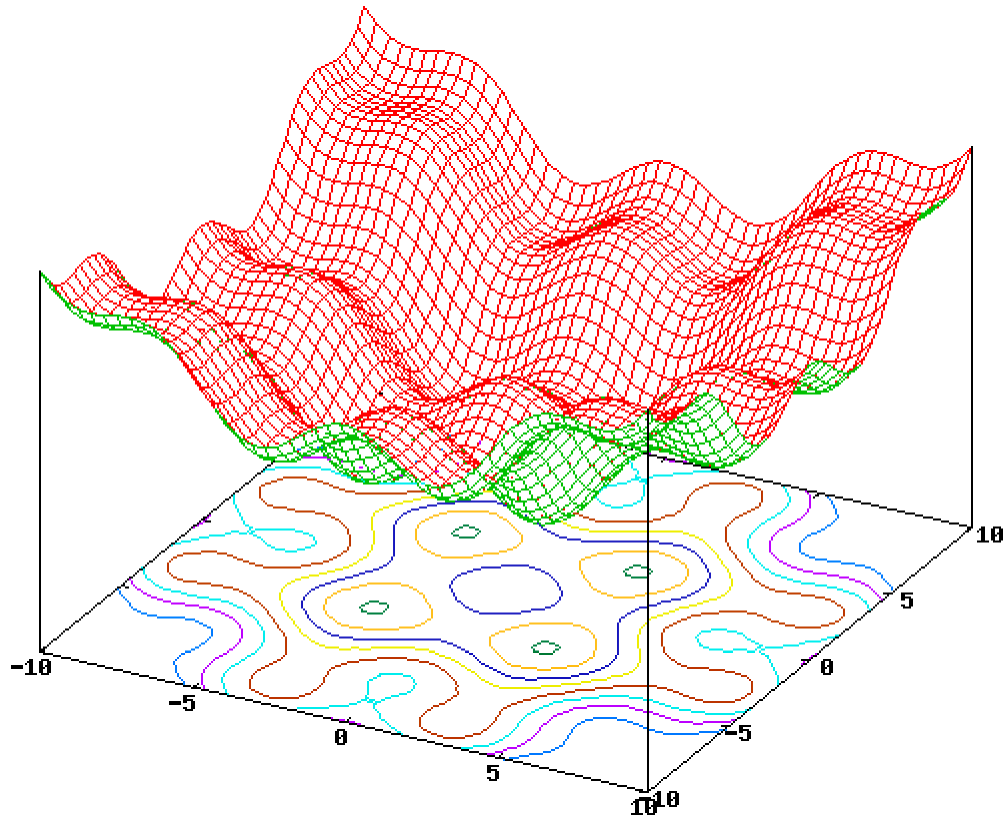
Parts 13

Global optimization

$$\begin{aligned} \text{minimize} \quad & f(x) \\ & g_i(x) = 0, \qquad i = 1, \ldots, n_e \\ & h_i(x) \geq 0, \qquad i = 1, \ldots, n_i \end{aligned}$$

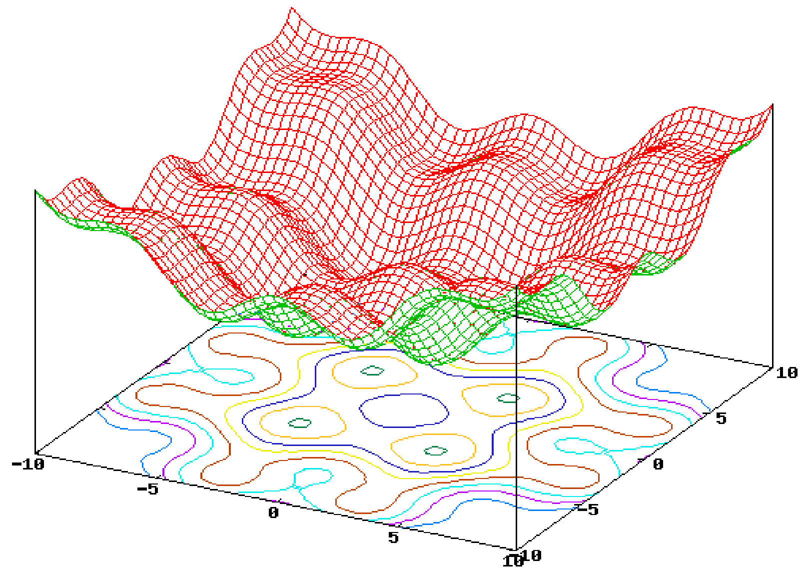
Motivation

What should we do when asked to find the (global) minimum of functions like this:



$$f(x) = \frac{1}{20}\left(x_1^2 + x_2^2\right) + \cos(x_1) + \cos(x_2)$$

A naïve sampling approach



A naïve approach: One way would be to sample at M-by-M points and choose the one with the smallest value.

Alternatively, we could start Newton's method at each of these points to get higher accuracy.

Problem: If we have n variables, then we would have to start at M^n points. This becomes prohibitive for large n!

Monte Carlo sampling

A better strategy ("Monte Carlo" sampling):

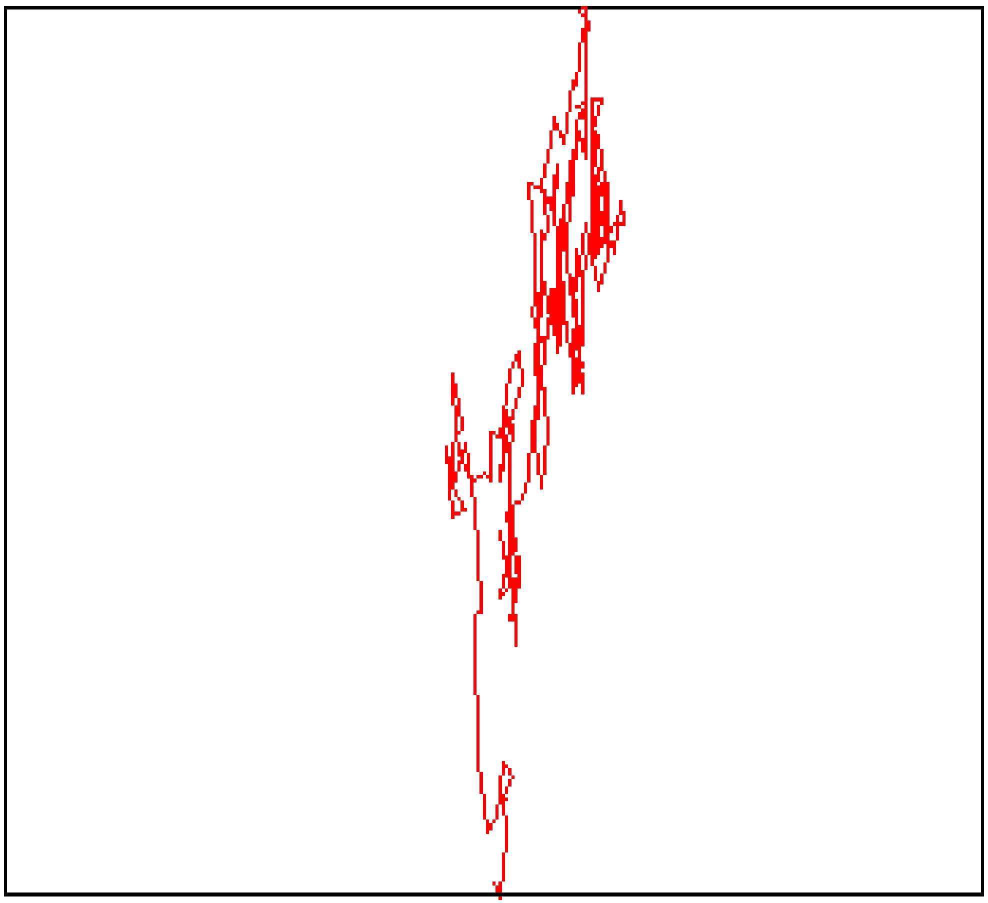
- Start with a feasible point x_0
- For *k=0,1,2,....*:
  - Choose a trial point x_t
  - If $f(x_t) \leq f(x_k)$ then $x_{k+1} = x_t$ [*accept* the sample]
  - Else:
    - . draw a random number s in *[0,1]*
    - . if $\exp\left[-\frac{f(x_t) - f(x_k)}{T}\right] \geq s$

      then $x_{k+1} = x_t$ [*accept* the sample]

      else $x_{k+1} = x_k$ [*reject* the sample]

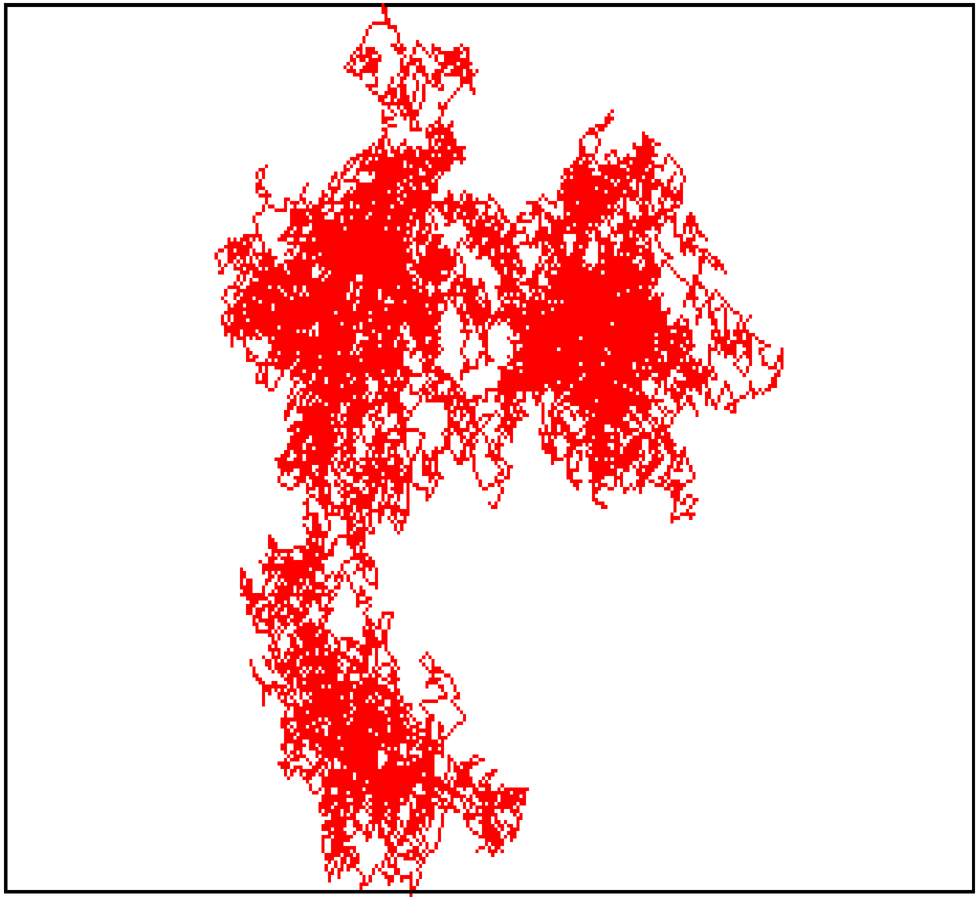
Monte Carlo sampling

Example: The first 200 sample points



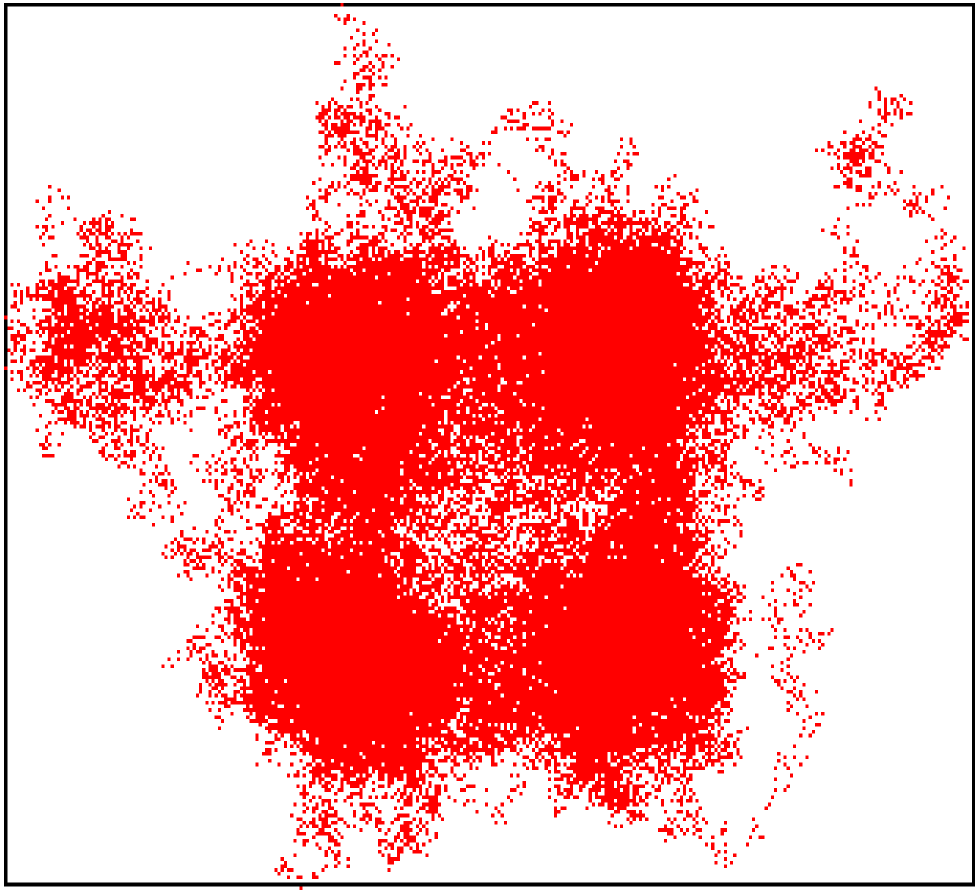
Monte Carlo sampling

Example: The first 10,000 sample points



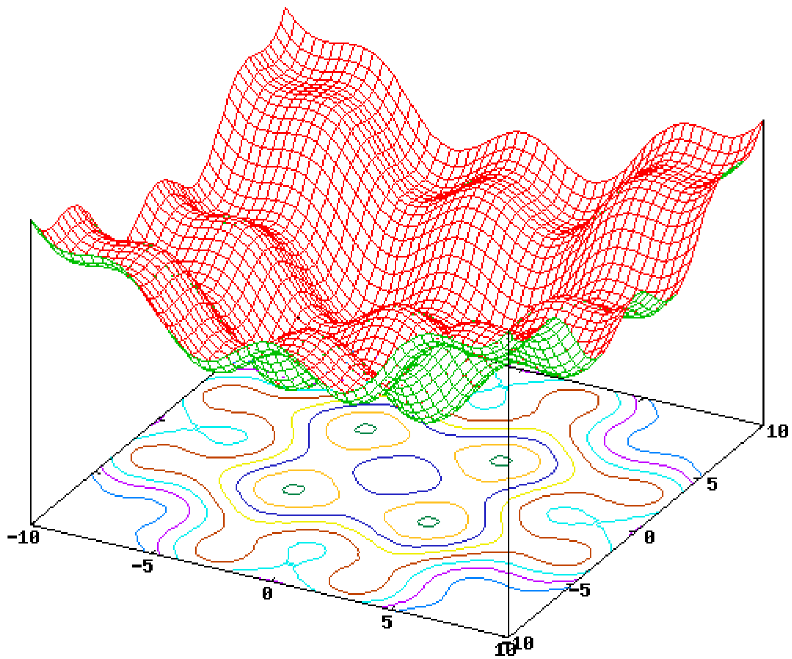
Monte Carlo sampling

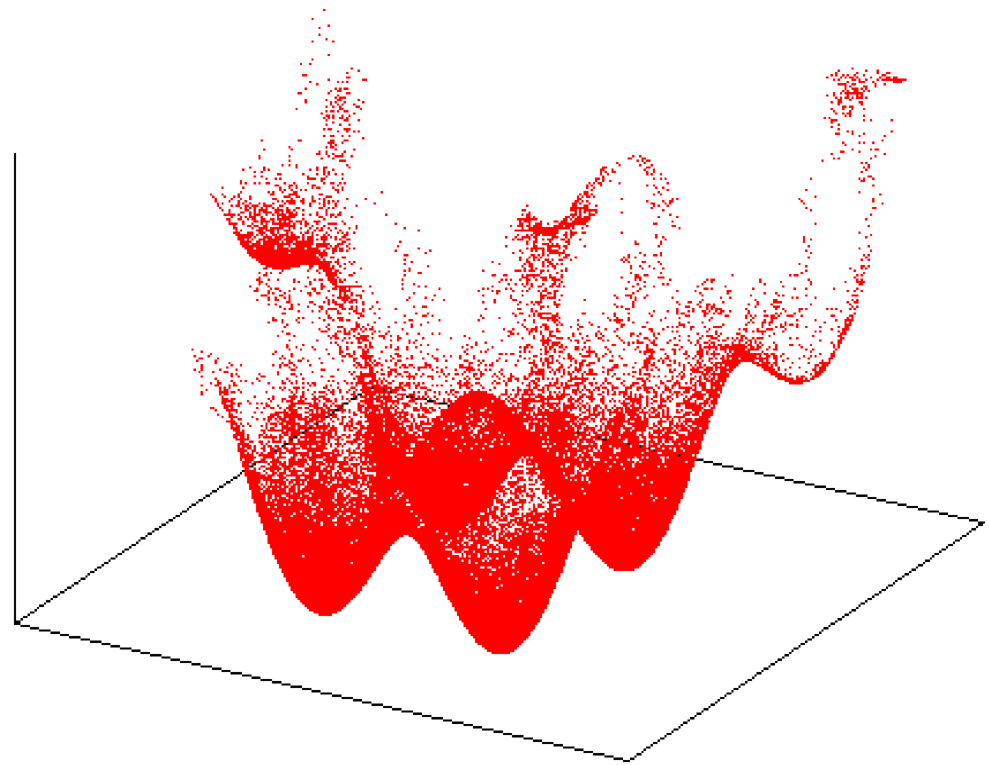
Example: The first 100,000 sample points



Monte Carlo sampling

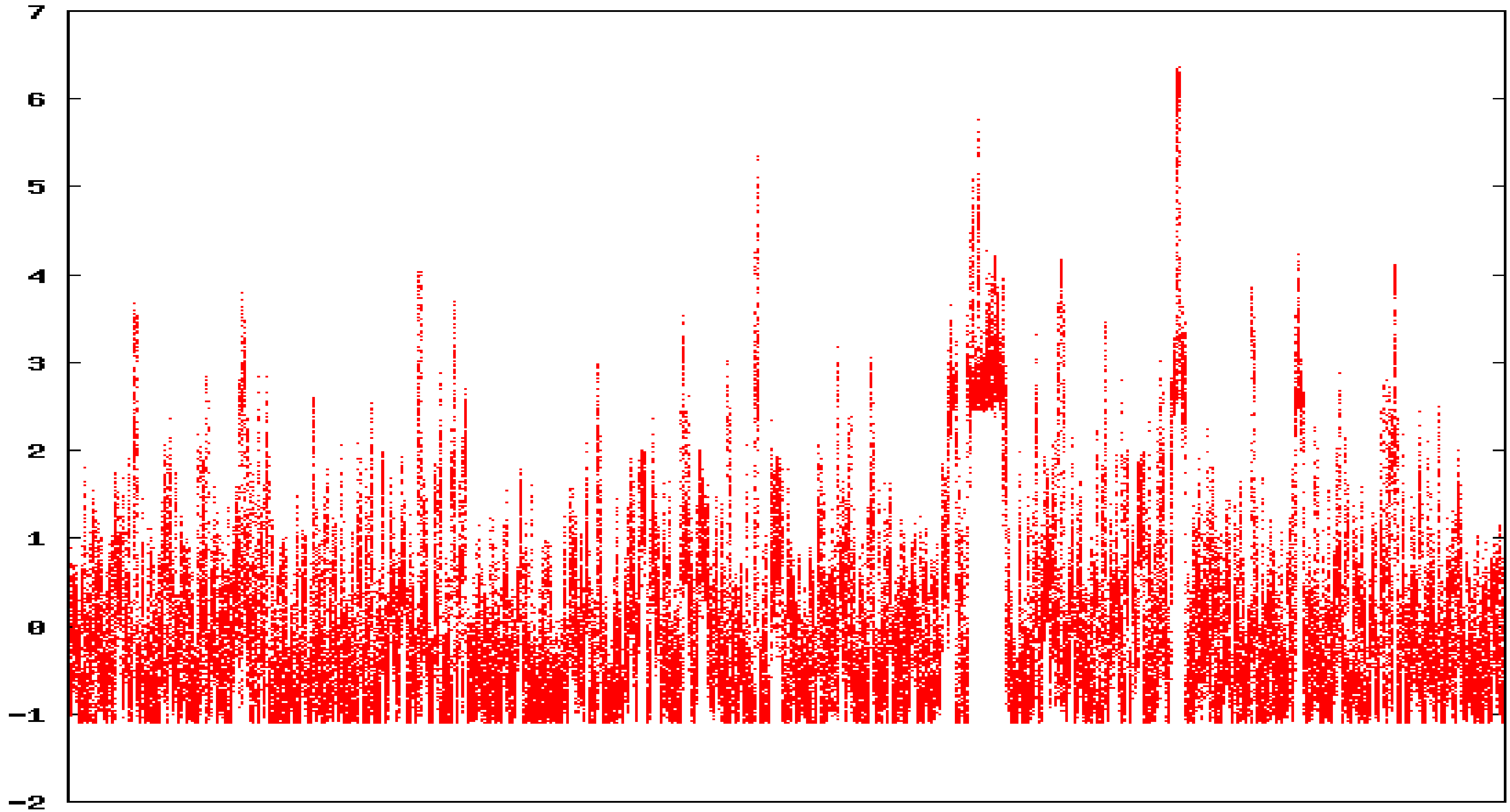
Example: Locations and values of the first 100,000 sample points





Monte Carlo sampling

Example: Values of the first 100,000 sample points



Note: The exact minimal value is -1.1032... . In the first 100,000 samples, we have 24 with values $f(x)<-1.103$.

How to choose the constant *T*:

- If T is chosen too small, then the condition

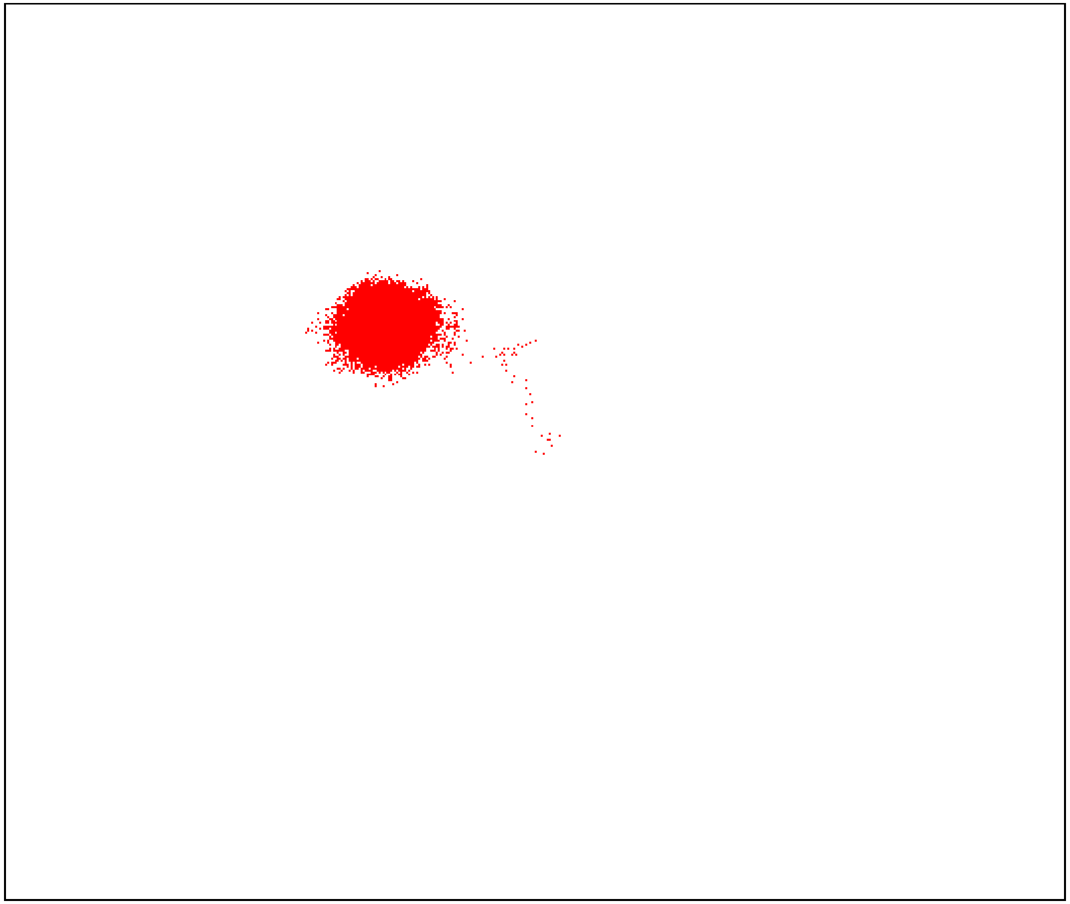
$$\exp\left[-\frac{f(x_t)-f(x_k)}{T}\right] \geq s, \qquad s \in U([0,1])$$

  will lead to frequent rejections of sample points for which *f(x)* increases.
  Consequently, we will get stuck in local minima for large numbers of samples before we accept a sequence of steps that gets "us over the hump".

- On the other hand, if T is chosen too large, then we will accept nearly every sample, irrespective of the value of *f(x)*.
  Consequently, we will perform a *random walk* that is no more efficient than uniform sampling.

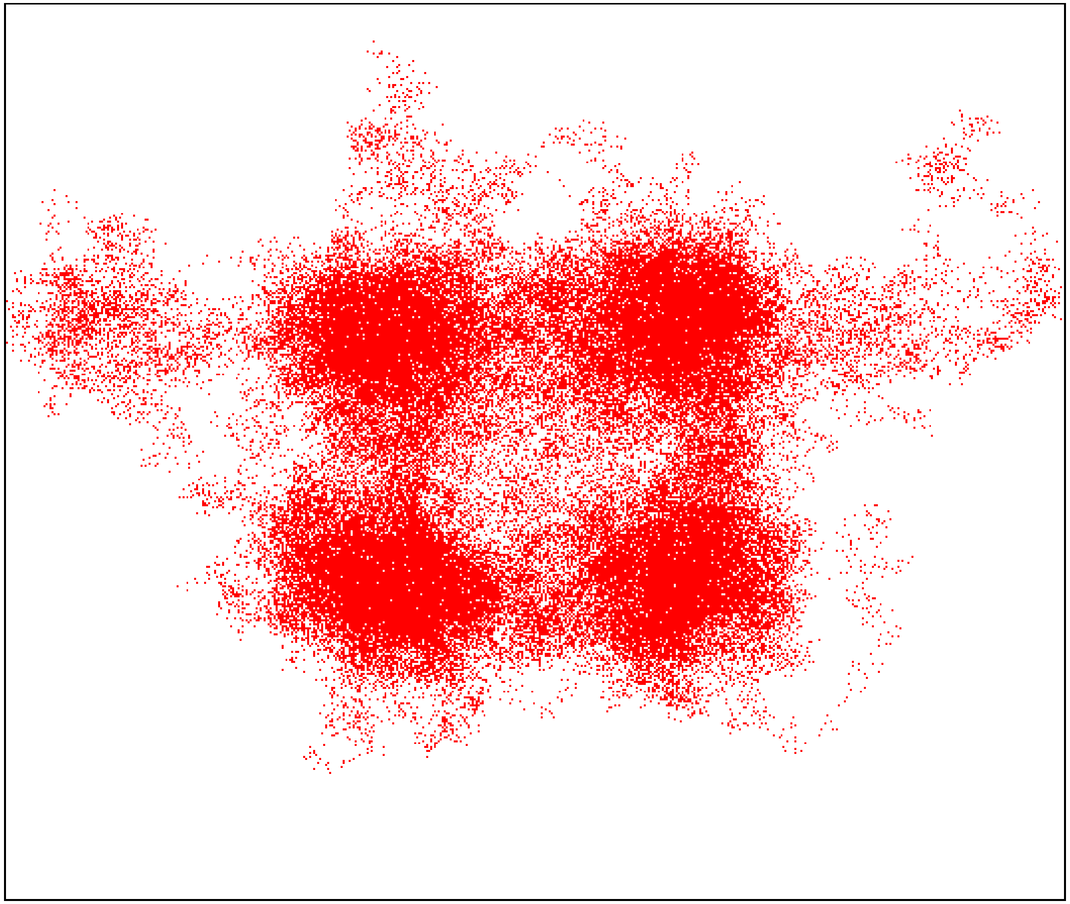
Monte Carlo sampling

Example: First 100,000 samples, T=0.1



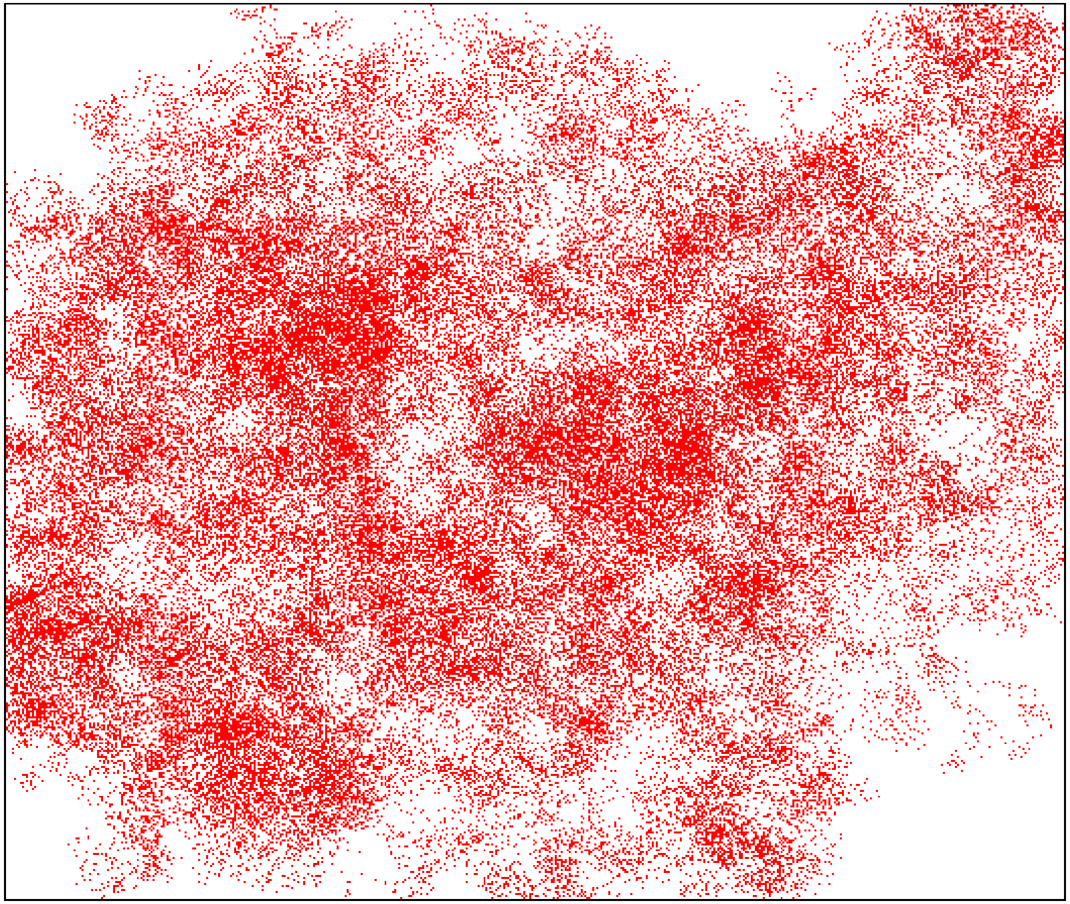
Monte Carlo sampling

Example: First 100,000 samples, T=1



Monte Carlo sampling

Example: First 100,000 samples, T=10



Monte Carlo sampling

Strategy: We need to choose T large enough that there is a reasonable probability to get out of local minima, but small enough that this doesn't happen before we have explored the region around a minimum.

Example: For $$f(x) = \frac{1}{20}\left(x_1^2 + x_2^2\right) + \cos(x_1) + \cos(x_2)$$
the difference in function value between local minima and saddle points is around 2. We want to choose T so that
$$\exp\left[-\frac{\Delta f}{T}\right] \geq s, \qquad s \in U([0,1])$$

is true maybe 10% of the time.

This is the case for $T=0.87$.

Monte Carlo sampling

How to choose the next sample x_t:

- If x_t is chosen independently of x_k then we just sample the entire domain, without exploring areas where *f(x)* is small. Consequently, we should choose x_t "close" to x_k.

- If we choose x_t too close to x_k we will have a hard time exploring a significant part of the feasible region.

- If we choose x_t in an area around x_k that is too large, then we don't adequately explore areas where *f(x)* is small.

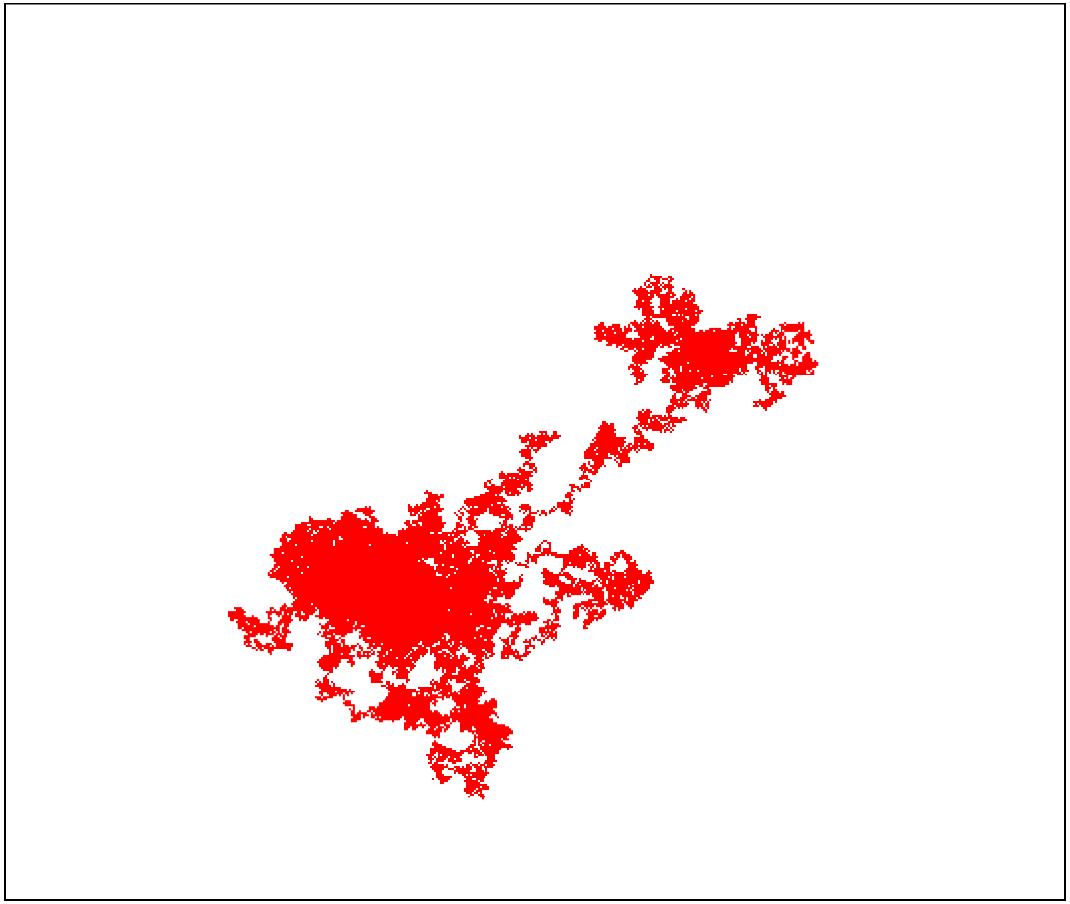
Common strategy: Choose

$$x_t = x_k + \sigma y, \qquad y \in N(0, I) \text{ or } U([-1,1]^n)$$

where σ is a fraction of the diameter of the domain or the distance between local minima.

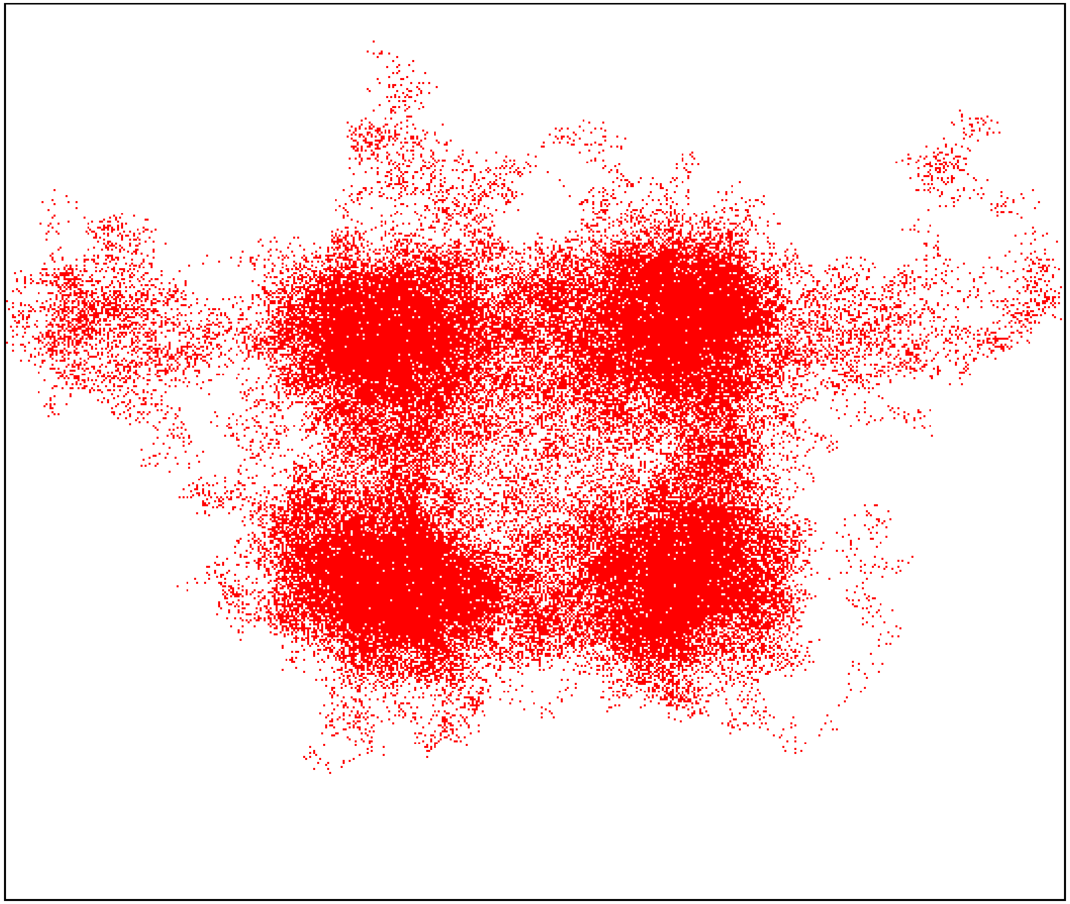
Monte Carlo sampling

Example: First 100,000 samples, T=1, σ=0.05



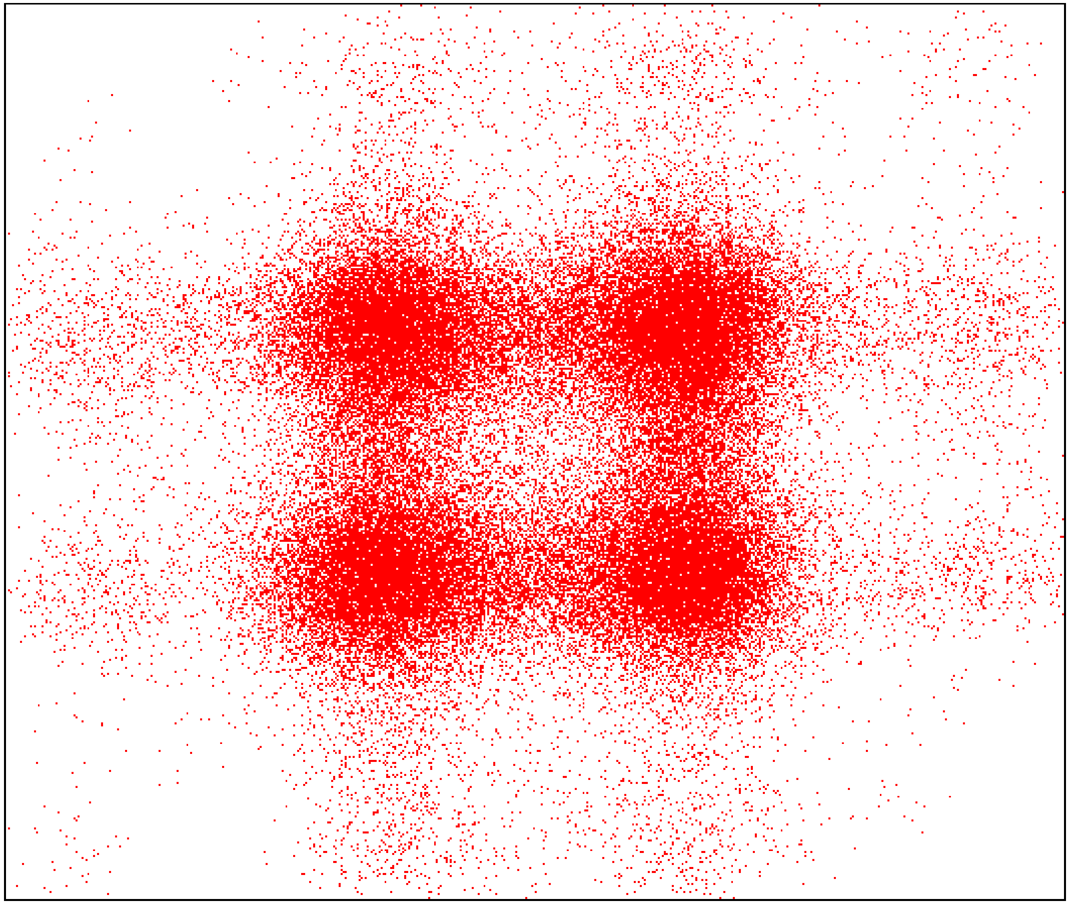
Monte Carlo sampling

Example: First 100,000 samples, T=1, σ=0.25



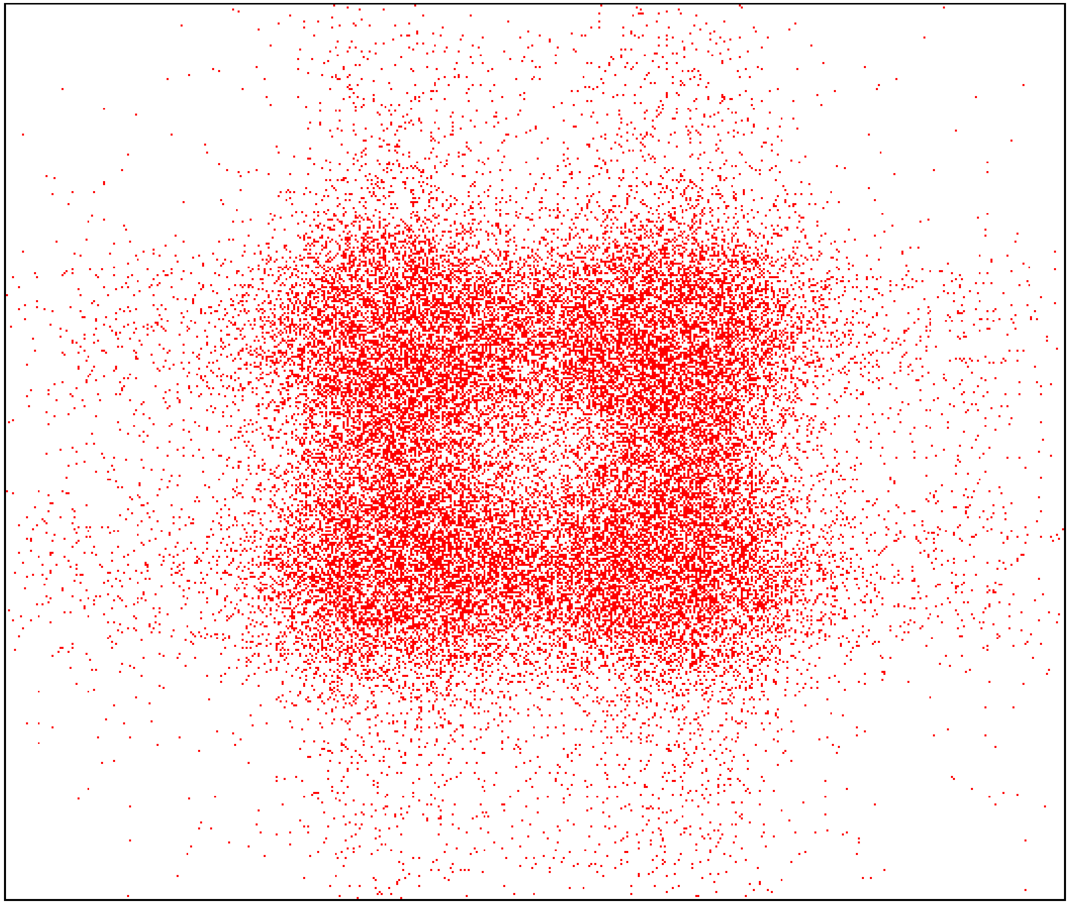
Monte Carlo sampling

Example: First 100,000 samples, T=1, σ=1



Monte Carlo sampling

Example: First 100,000 samples, T=1, σ=4



Monte Carlo sampling with constraints

Inequality constraints:

- In case of simple inequality constraints, one can modify the sample generation strategy to never generate trial samples that don't satisfy the inequalities
- For complex inequality constraints, always reject samples for which

$$h_i(x_t) < 0 \qquad \text{for at least one } i$$

Monte Carlo sampling with constraints

Inequality constraints:

- In case of simple inequality constraints, one can modify the sample generation strategy to never generate trial samples that don't satisfy the inequalities
- This is equivalent to using the following acceptance strategy:

- If $Q(x_t) \leq Q(x_k)$ then $x_{k+1} = x_t$

- Else:

. draw a random number s in *[0,1]*

. if

$$\exp\left[-\frac{Q(x_t) - Q(x_k)}{T}\right] \geq s$$

then

$$x_{k+1} = x_t$$

else

$$x_{k+1} = x_k$$

where

$$Q(x) = \infty \text{ if at least one } h_i(x) < 0, \qquad Q(x) = f(x) \text{ otherwise}$$

Monte Carlo sampling with constraints

Equality constraints:

- We need to generate only samples that satisfy equality constraints

- If we have only linear equality constraints of the form

$$g(x) = Ax - b = 0$$

then one way to guarantee this is to generate samples using

$$x_t = x_k + \sigma Z y, \qquad y \in \mathbb{R}^{n-n_e}, \quad y = N(0, I) \text{ or } U([-1,1]^{n-n_e})$$

where Z is the null space matrix of A, i.e. $AZ=0$.

Monte Carlo sampling

Theorem:

Let A be a subset of the feasible region. Let certain conditions on the sample generation strategy hold. Then in the limit $k \to \infty$ we have

$$\text{number of samples } x_k \in A \;\propto\; \int_A e^{-\frac{f(x)}{T}}\,dx$$

In other words, we are guaranteed that every region A will be adequately sampled over time, and that the area around the global minimum will be better sampled than any other region.

In particular,

$$\text{fraction of samples } x_k \in A \;=\; \frac{1}{C}\int_A e^{-\frac{f(x)}{T}}\,dx + O\left(\frac{1}{\sqrt{N}}\right)$$

Monte Carlo sampling

Remark:

Monte Carlo sampling appears to be a strategy that bounces around randomly, taking into account only the values (not the derivatives) of the objective function $f(x)$.

However, that is not so if the sample generation strategy and T are chosen carefully: In that case, we choose a new sample moderately close to the previous one, and we always accept it if $f(x)$ is reduced, whereas we only sometimes accept it if $f(x)$ is increased by this step.

In other words, on average we still move in the direction of steepest descent!

Simulated Annealing

Motivation:
Particles in a gas, or atoms in a crystal have an energy that is on average in equilibrium with the rest of the system. At any given time, however, its energy may be higher or lower.

In particular, the probability that its energy is E is

$$P(E) \propto e^{-\frac{E}{k_B T}}$$

Where k_B is the Boltzmann constant. Likewise, the probability that a particle can overcome an energy barrier of height ΔE is

$$P(E \to E + \Delta E) \propto \min\left\{1, e^{-\frac{\Delta E}{k_B T}}\right\} = \left\{\begin{array}{c} 1 \text{ if } \Delta E \leq 0 \\ e^{-\frac{\Delta E}{k_B T}} \text{ if } \Delta E > 0 \end{array}\right\}$$

This is exactly the Monte Carlo transition probability if we identify

$$E = f\, k_B$$

Simulated Annealing

Motivation:

In other words, Monte Carlo sampling is analogous to watching particles bounce around in a potential when driven by a gas at constant temperature.

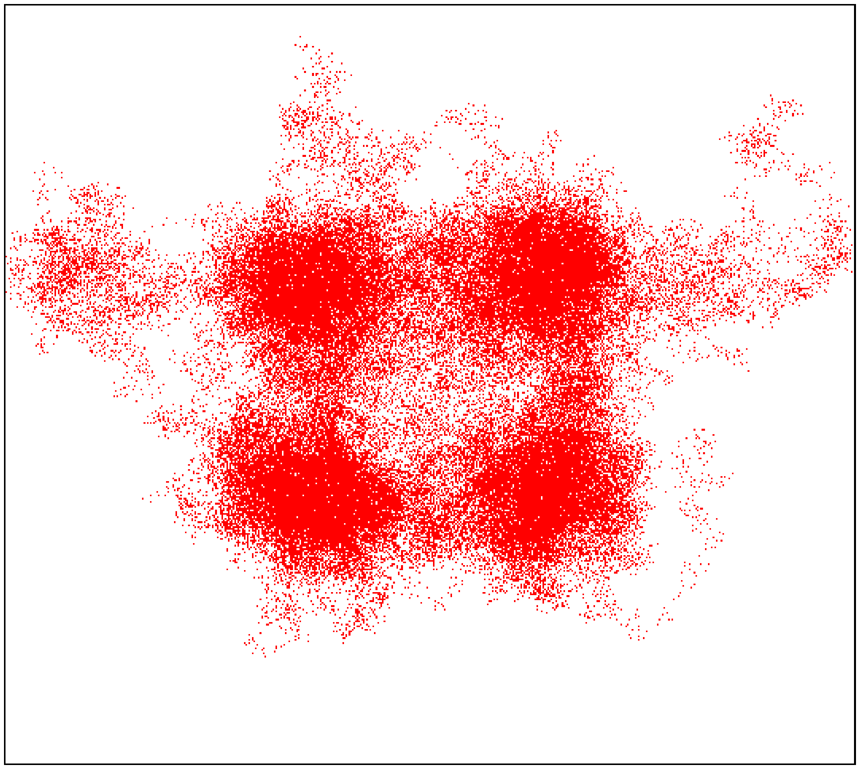
On the other hand, we know that if we slowly reduce the temperature of a system, it will end up in the ground state with very high probability. For example, slowly reducing the temperature of a melt results in a perfect crystal. (On the other hand, reducing the temperature too quickly results in a glass.)

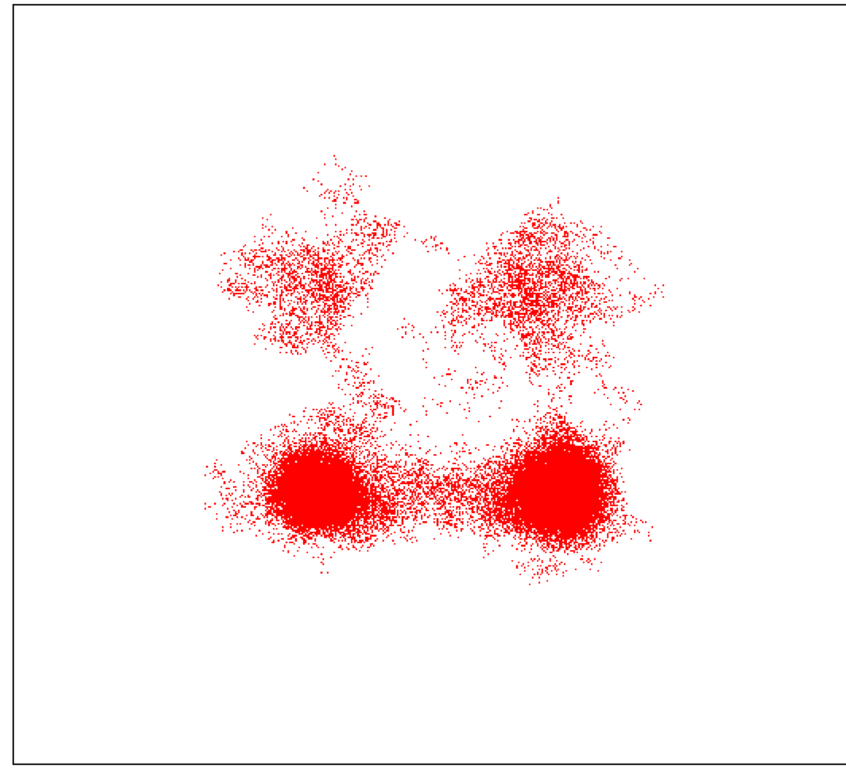
The *Simulated Annealing* algorithm uses this analogy by using the modified transition probability

$$\exp\left[-\frac{f(x_t)-f(x_k)}{T_k}\right] \geq s, \qquad s \in U([0,1]), \qquad T_k \to 0 \text{ as } k \to \infty$$

Simulated Annealing

Example: First 100,000 samples, σ=0.25



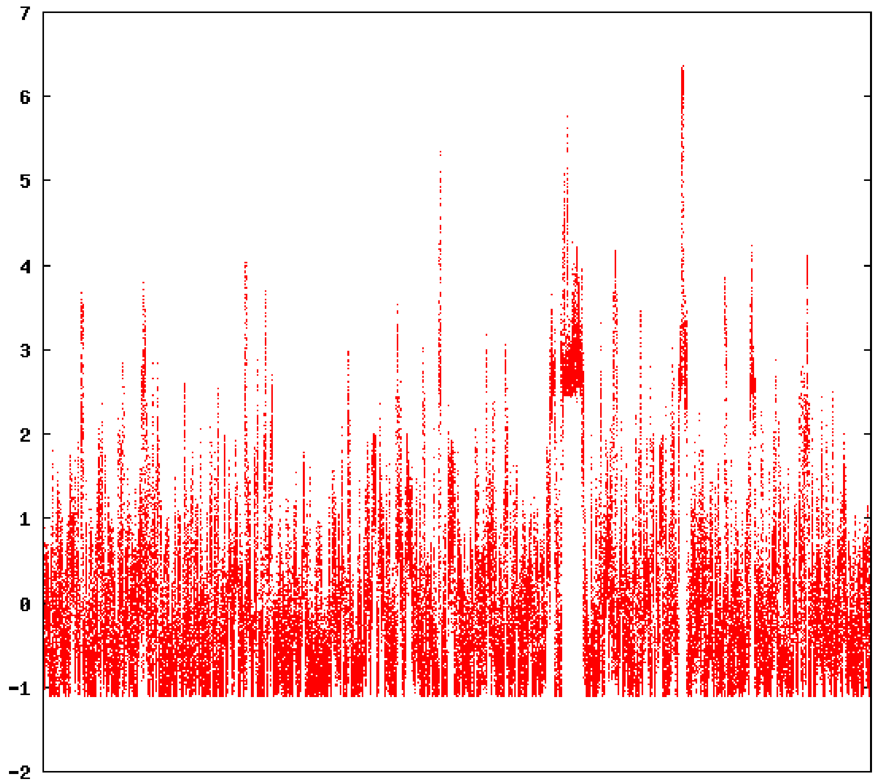


$T = 1$

$$T_k = \frac{1}{1 + 10^{-4} k}$$

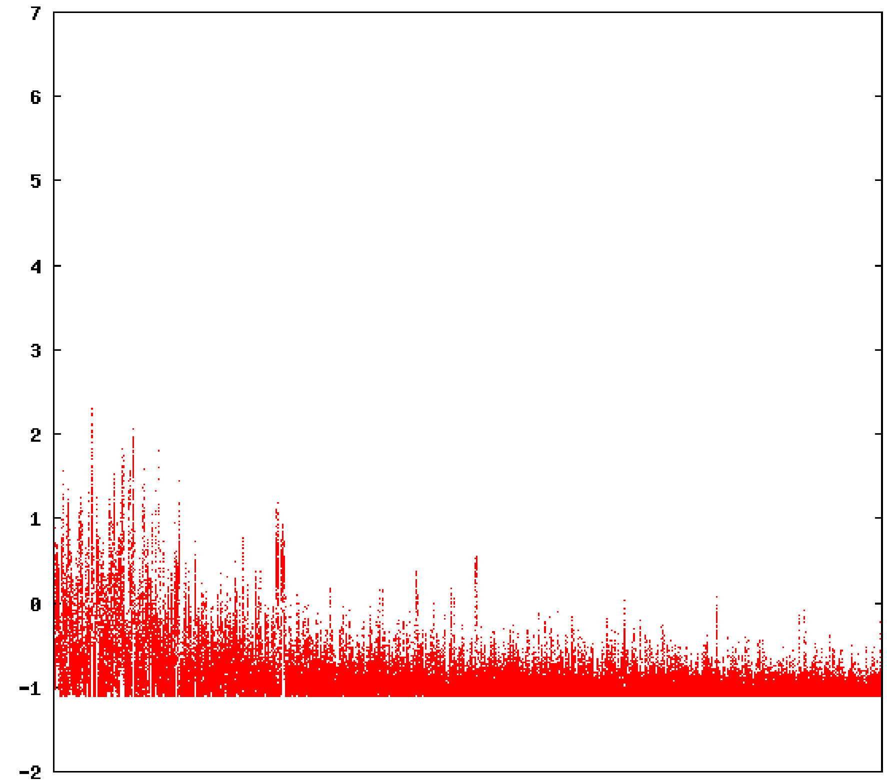
Simulated Annealing

Example: First 100,000 samples, σ=0.25



$T = 1$

24 samples with *f(x)<-1.103*

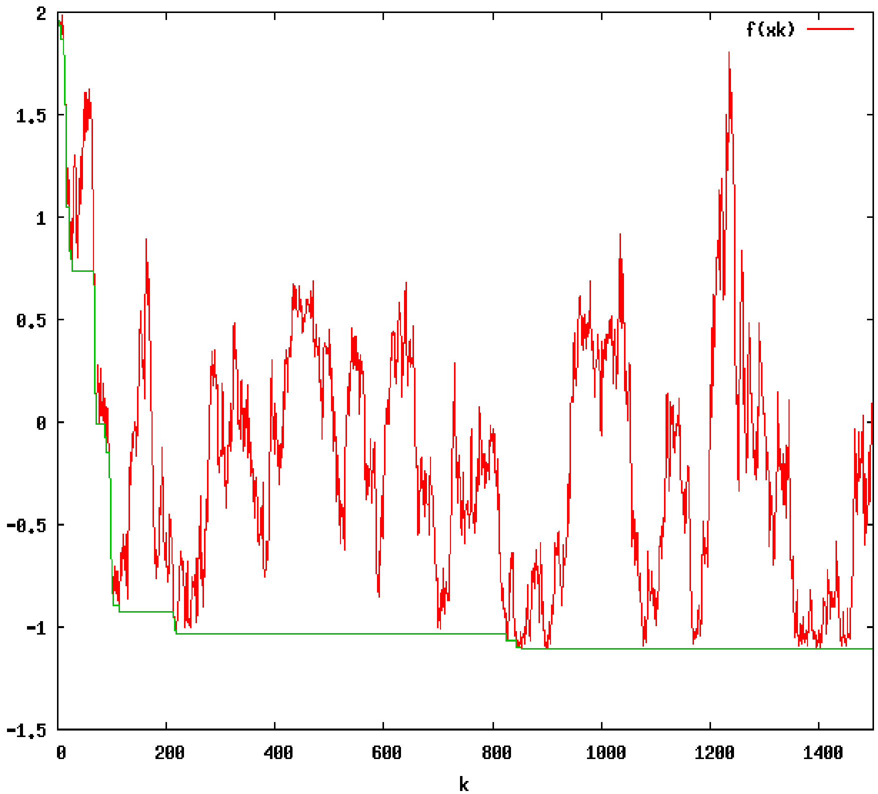


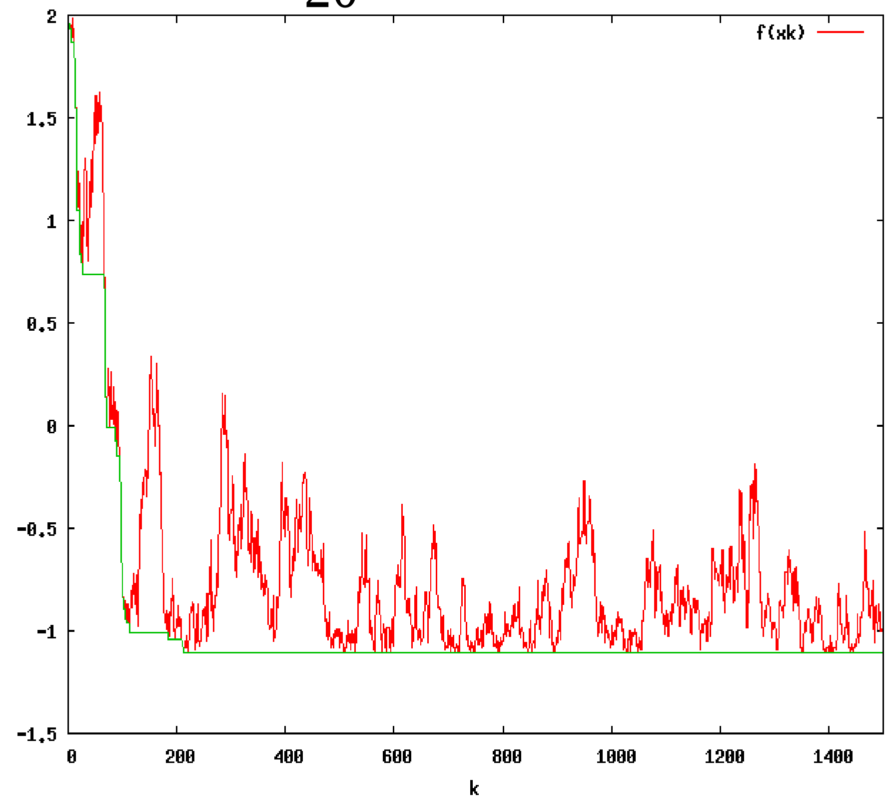
$$T_k = \frac{1}{1+10^{-4}k}$$

192 samples with *f(x)<-1.103*

Simulated Annealing

Convergence: First 1,500 samples, $f(x) = \sum_{i=1}^{2} \frac{1}{20} x_i^2 + \cos(x_i)$





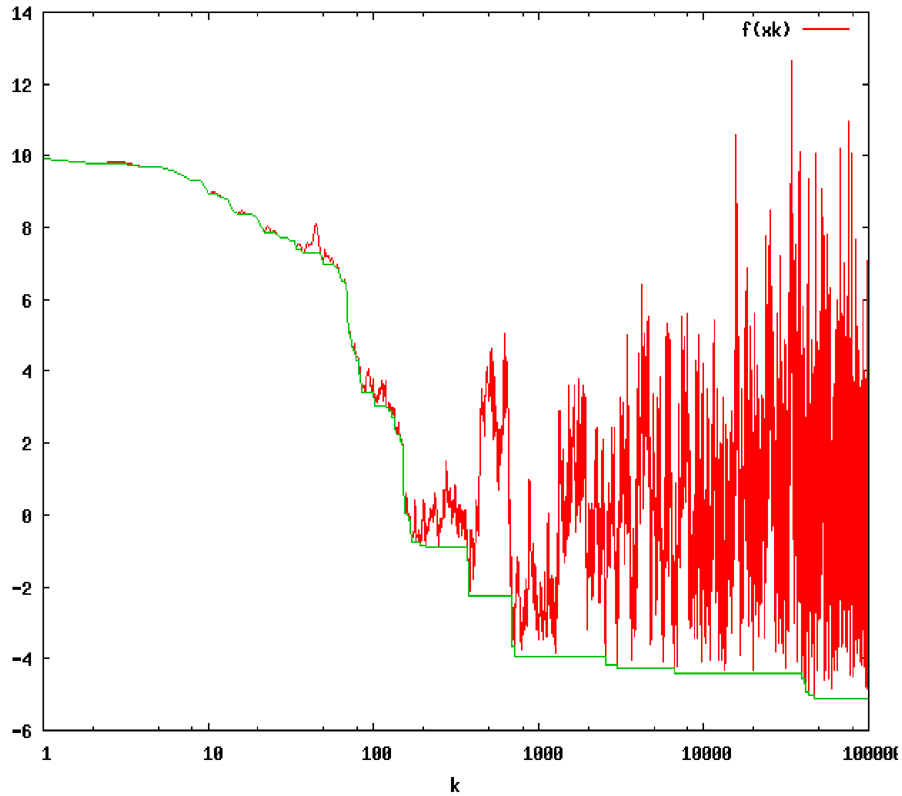
$T = 1$

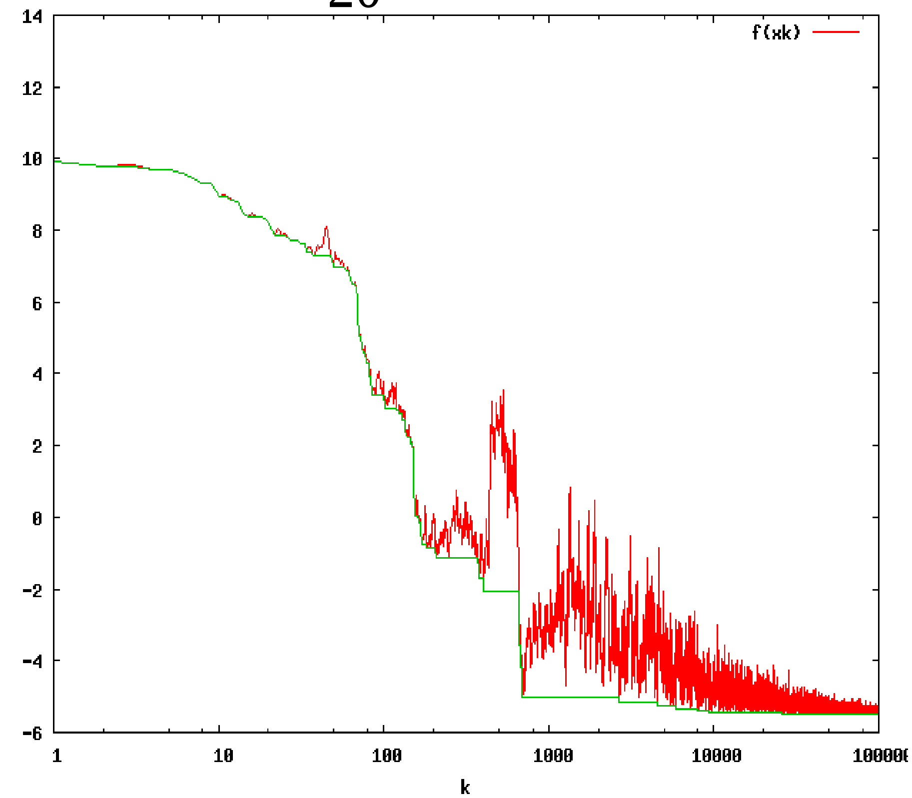
$T_k = \frac{1}{1 + 0.005\,k}$

(Green line indicates the lowest function value found so far)

Simulated Annealing

Convergence: First 10,000 samples, $f(x) = \sum_{i=1}^{10} \frac{1}{20} x_i^2 + \cos(x_i)$





$T = 1$

$$T_k = \frac{1}{1 + 0.0005k}$$

(Green line indicates the lowest function value found so far)

Simulated Annealing

Discussion:

Simulated Annealing is often more efficient in finding global minima because it initially explores the energy landscape at large, and later on explores the areas of low energy in greater detail.

On the other hand, there is now another knob to play with (namely how we reduce the temperature):

- If the temperature is reduced too fast, we may get stuck in local minima (the "glass" state)
- If the temperature is not reduced fast enough, the algorithm is no better than Monte Carlo sampling and may require many many samples.

Very Fast Simulated Annealing (VFSA)

A further refinement:
In *Very Fast Simulated Annealing* we not only reduce temperature over time, but also reduce the search radius of our sample generation strategy, i.e. we compute

$$x_t = x_k + \sigma_k\, y, \qquad y \in N(0, I) \text{ or } U([-1,1]^n)$$

and let

$$\sigma_k \to 0$$

Like reducing the temperature, this ensures that we sample the vicinity of minima better and better over time.

Remark: To guarantee that the algorithm can reach any point in the search domain, we need to choose σ_k so that

$$\sum_{k=0}^{\infty} \sigma_k = \infty$$

Genetic Algorithms (GA)

An entirely different idea:

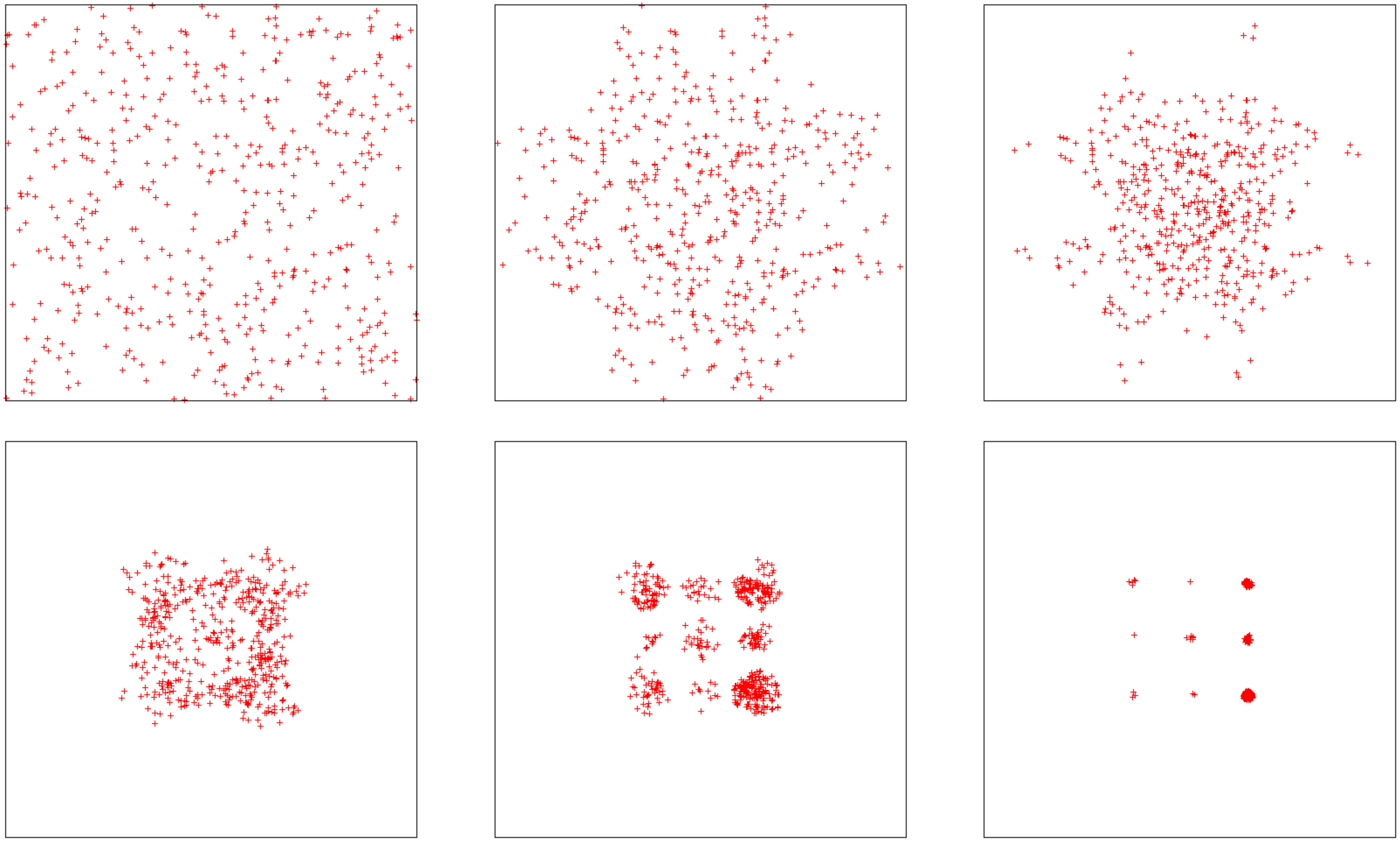
Choose a set ("population") of N points ("individuals") $P_0=\{x_1,\ldots x_N\}$

For $k=0,1,2,\ldots$ ("generations"):

- Copy those $N_f<N$ individuals in P_k with the smallest $f(x)$ (i.e. the "fittest individuals") into P_{k+1}
- While $\#P_{k+1}<N$:
  - select two individuals ("parents") x_a, x_b from among the first N_f individuals in P_{k+1} with probabilities proportional to $e^{-f(x_i)/T}$
  - create a new point x_{new} from x_a, x_b ("mating")
  - perform some random changes on x_{new} ("mutation")
  - add it to P_{k+1}

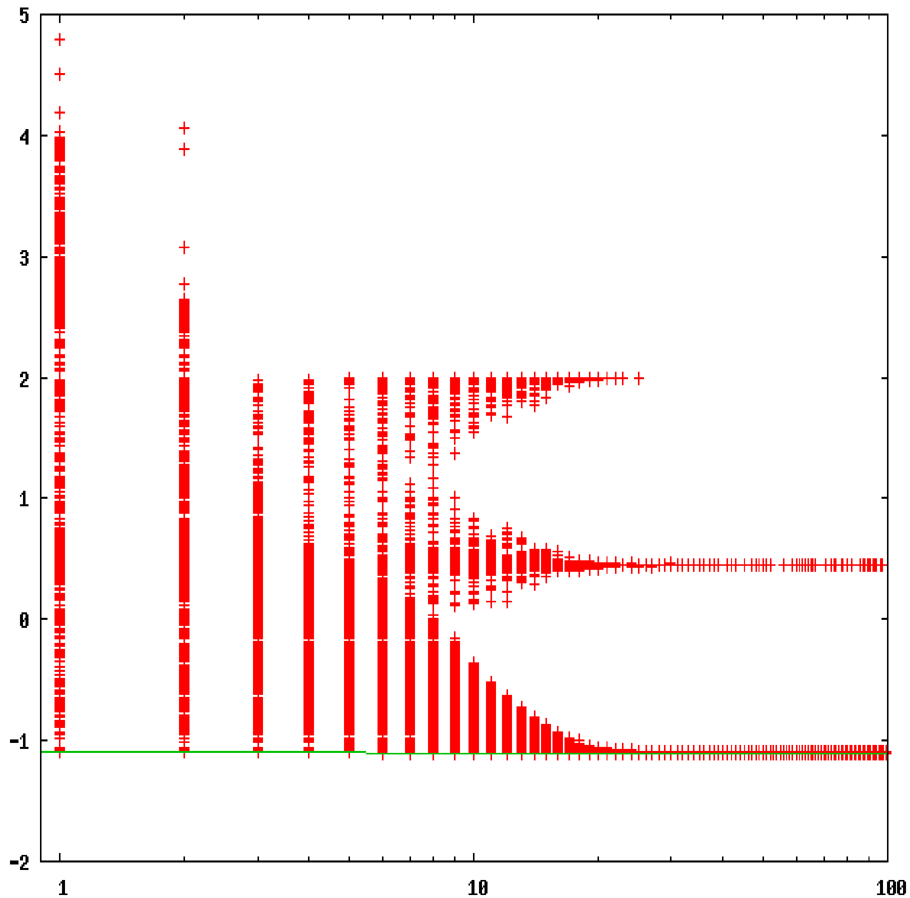
Genetic Algorithms (GA)

Example: Populations at $k=0,1,2,5,10,20$, $N=500$, $N_s=2/3\ N$

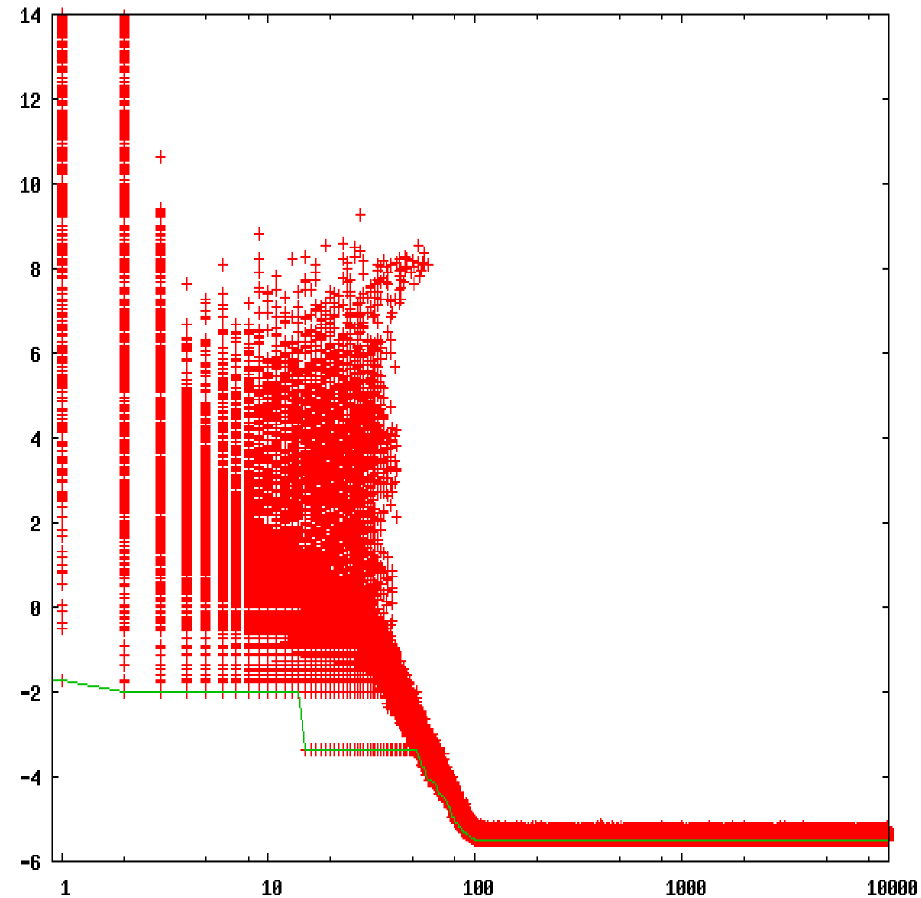


Genetic Algorithms (GA)

Convergence: Values of the N samples for all generations k



$$f(x)=\sum_{i=1}^{2}\frac{1}{20}x_i^2+\cos(x_i)$$



$$f(x)=\sum_{i=1}^{10}\frac{1}{20}x_i^2+\cos(x_i)$$

Genetic Algorithms (GA)

Mating:

- Mating is meant to produce new individuals that share the traits of the two parents
- If the variable *x* encodes real values, then mating could just take the mean value of the parents:

$$x_{new} = \frac{x_a + x_b}{2}$$

- For more general properties (paths through cities, which of *M* objects to put where in a suitcase, …) we have to encode *x* in a binary string. Mating may then select bits (or bit sequences) randomly from each of the parents

- There is a huge variety of encoding and selection strategies in the literature.

Mutation

- Mutations are meant to introduce an element of randomness into the process, to explore search directions that aren't represented yet in the population
- If the variable *x* represents real values, we can just add a small random value to *x* to simulate mutations

$$x_{new} = \frac{x_a + x_b}{2} + \epsilon y, \quad y \in \mathbb{R}^n, \quad y = N(0, I)$$

- For more general properties, mutations can be introduced by randomly flipping individual bits or bit sequences in the encoded properties

- There is a huge variety of mutation strategies in the literature.

Part 13

Summary of global optimization methods

$$
\begin{aligned}
\text{minimize} \quad & f(x) \\
& g_i(x) = 0, \qquad i = 1, \ldots, n_e \\
& h_i(x) \geq 0, \qquad i = 1, \ldots, n_i
\end{aligned}
$$

Summary of methods

- Global optimization problems with many minima are difficult because of the curse of dimensionality: the number of places where a minimum could be becomes very large if the number of dimensions becomes large
- There is a large zoo of methods for these kinds of problems
- Most algorithms use some sort of stochasticity to sample the feasible region
- Algorithms also work for non-smooth problems
- Most methods are not very effective (if one counts number of function evaluations) in return for the ability to get out of local minima

- Global optimization algorithms should *never* be used whenever we know that the problem has only a small number of minima and/or is smooth and convex