深蓝学院
shenlanxueyuan.com

# Numerical Optimization in Robotics

## Lecture 1:

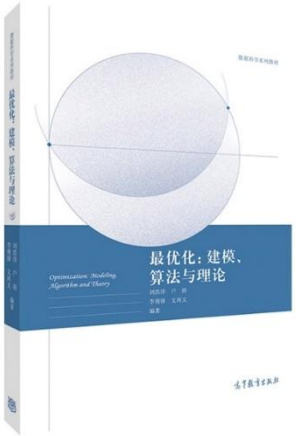
## Preliminaries

主讲人 Zhepei Wang

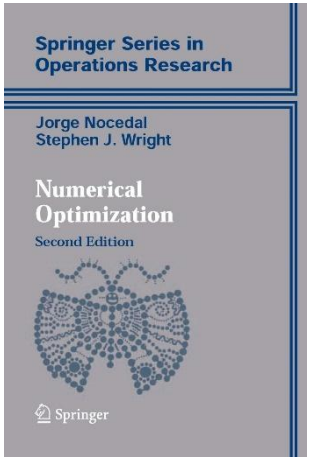
Ph.D. in Robotics
Zhejiang University

# Introduction

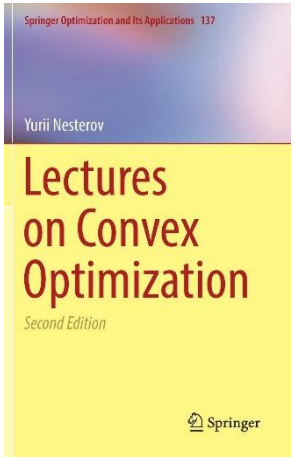
# Recommended Reading List



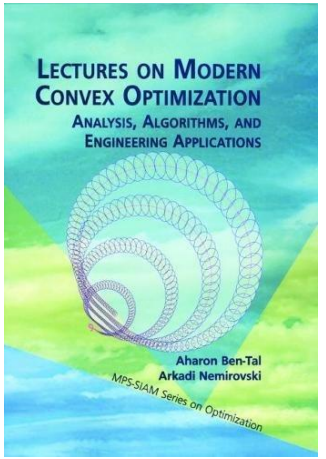

Friendly to Chinese students




Numerical implementation




Convex Optimization Theory




Conic Programming

# Mathematical Optimization

optimization problem

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

$x = (x_1, \cdots, x_n) \in \mathbb{R}^n$: optimization variables

$f$: $\mathbb{R}^n \mapsto \mathbb{R}$ : objective function

$g$: $\mathbb{R}^n \mapsto \mathbb{R}^{m_g}$ : inequality constraint functions

$h$: $\mathbb{R}^n \mapsto \mathbb{R}^{m_h}$: equality constraint functions

**optimal solution** $x^*$ has smallest value $f$ among all vectors that satisfy the constraints.

# Mathematical Optimization

optimization problem

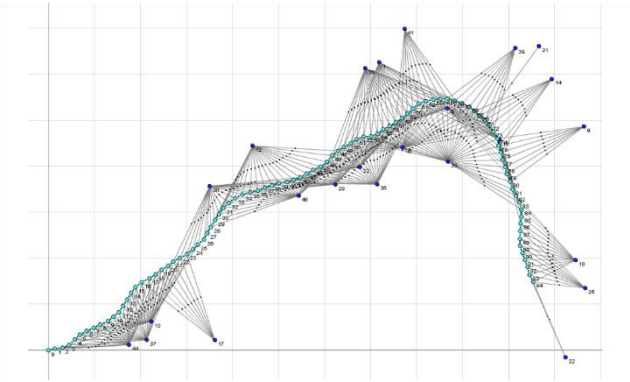
$$\begin{aligned} \min \ & f(x) \\ \text{s.t.} \ & g(x) \leq 0 \\ & h(x) = 0 \end{aligned}$$

default assumptions:

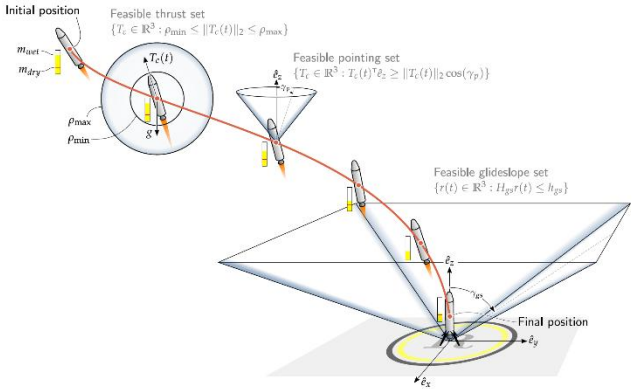
1. The objective function is lower bounded.
2. The objective function has bounded sub-level sets.

# Optimization in Robotics

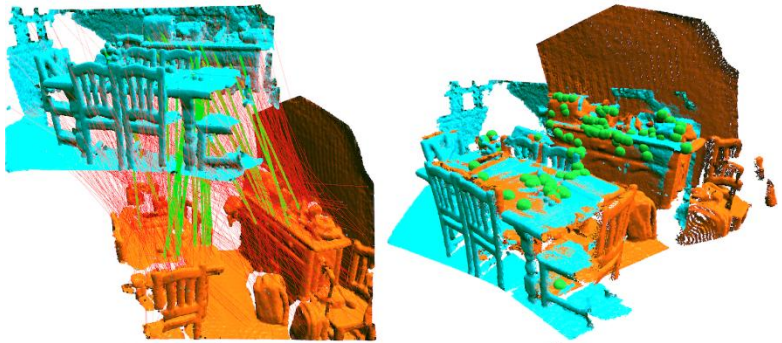
Smooth and Mapping: Nonlinear Least Squares



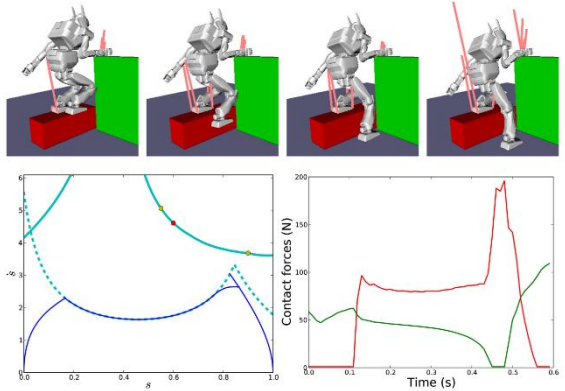
Trajectory Planning: Nonlinear Program




Registration: Semi-Definite Program




Time-Optimal Path Parameterization: Second-Order Conic Program



# Course Topics

## 1st Course: Fundamentals of Numerical Optimization

1. Mathematical Programming and Robotics
2. Convexity in Nonconvex Optimization
3. Convex Sets and Convex Functions
4. Unconstrained Nonconvex Optimization
5. Line-Search Steepest Gradient Descent
6. Modified Damped Newton's Method

Nonconvex Problems

## 2nd Course: Unconstrained Optimization

1. Quasi-Newton Methods
2. BFGS Update
3. Strong/Weak Curvature Conditions
4. Li-Fukushima Global Convergence Condition
5. Limited Memory BFGS Update
6. Lewis-Overton Line Search for Non-Smooth Functions
7. Linear Conjugate Gradient Descent
8. Newton-CG Method
9. Application: Smooth Navigation Path Generation

Nonconvex Problems

# Course Topics

Nonconvex Problems

## 3rd Course: Constrained Optimization

1. Category and Complexity of Constrained Optimization
2. Low-Dimensional Linear Program: Seidel's LP
3. Low-Dimensional Strictly Convex Quadratic Program
4. Sequential Unconstrained Minimization Technique (SUMT)
5. SUMT: Penalty Method and Barrier Method
6. SUMT: Lagrangian Relaxation and Uzawa's Method
7. Karush-Kuhn-Tucker Conditions
8. PHR  Augmented Lagrangian Method
9. Application 1: Control Allocation Problem
10. Application 2: Collision Distance Computation
11. Application 3: Nonlinear Model Predictive Control

## 4th Course: Optimization for Symmetric Cones

1. Cones and Symmetric Cones
2. Symmetric Cones and Euclidean Jordan Algebra
3. Representability of Cones
4. Augmented Lagrangian for Symmetric Cones
5. Semi-Smooth Newton Method
6. Application: Time-Optimal Path Parameterization

Convex Problems

## 5th Course: Techniques in Formulating and Solving Problems

1. Smoothing Techniques
2. Degrees of Freedom and Adjoint Method
3. Category and Features of Linear Solvers
4. Project: Enabling Safe Navigation in Dense-Obstacle Environments

Nonconvex Problems

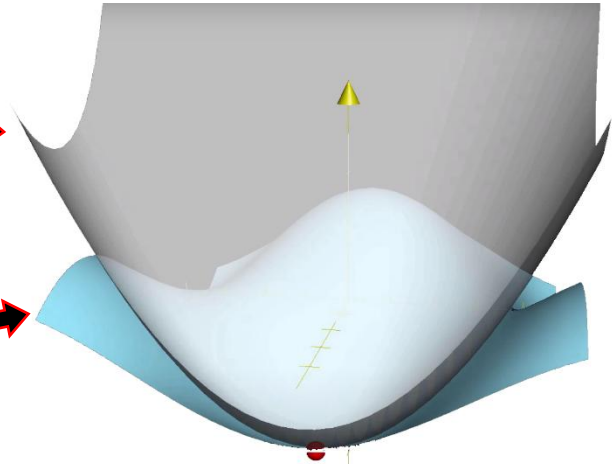
# Convexity in Nonconvex Optimization

# Convexity in Nonconvex Optimization

Why do we need to know convexity?

Convex function

Nonconvex function



- Optimization of convex functions over convex sets is well studied.
- Optimization algs utilize properties of convex fun/set to analyze the convergence.
- Some important problems have convex formulation/relaxiation.
- Many nonconvex functions are locally convex near the interested local minima.
- In practice, local minima of nonconvex functions can be good enough.

# Convex Sets

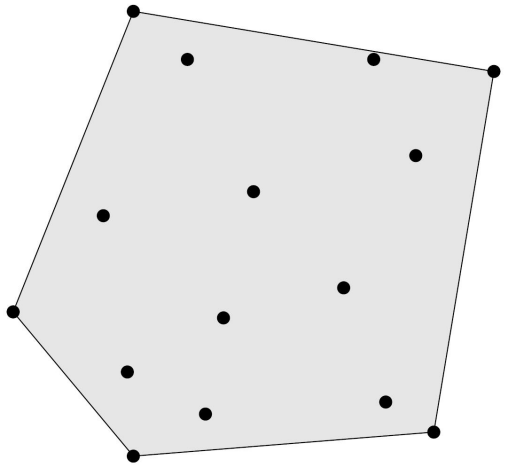
# Convex Sets

Def: A set is convex if every line between its two points stays in the set?

$$\theta x_1 + (1-\theta)x_2, \quad 0 \leq \theta \leq 1$$

More General:
All **convex combinations**
lie in set.

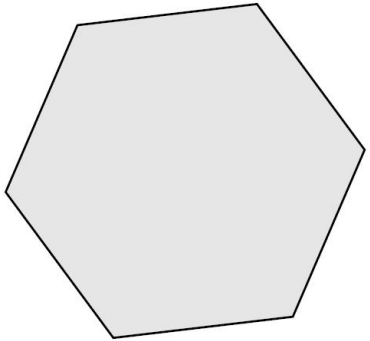
$$\sum \theta_i x_i = \theta_1 x_1 + \theta_2 x_2 + \theta_3 x_3$$
$$\sum \theta_i = \theta_1 + \theta_2 + \theta_3 = 1, \quad \theta_i \geq 0 \forall i$$

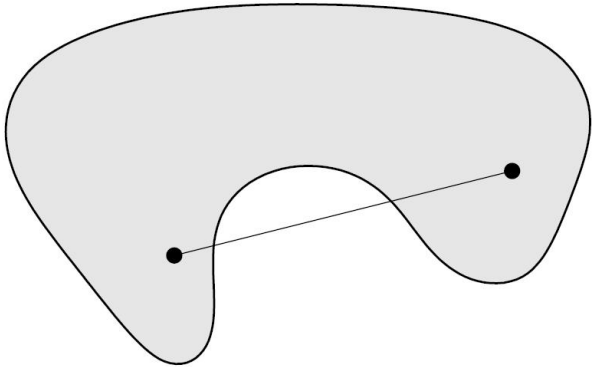


These definitions are the same!

# Convex Sets

Is it convex?

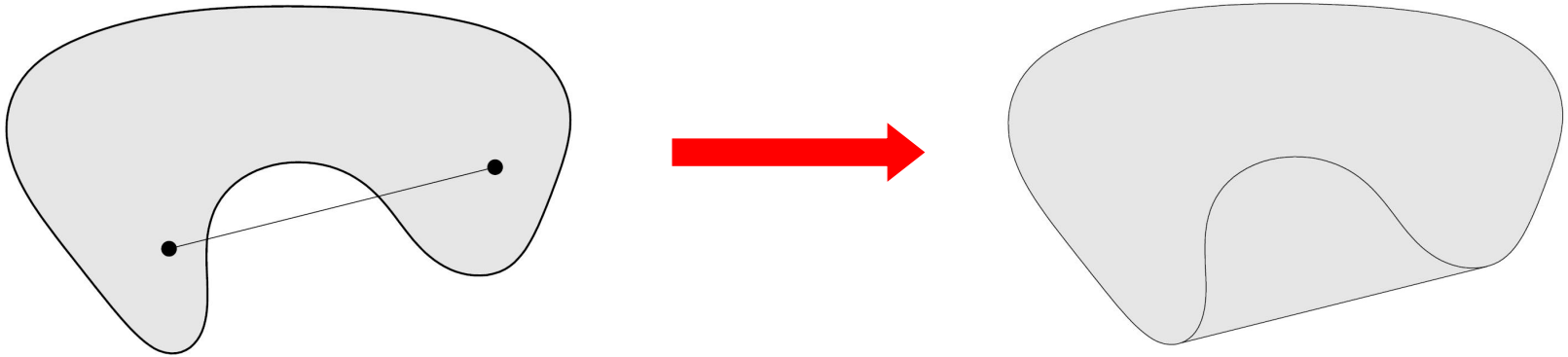




What if the square was round?

# Convex Sets

Convex hull: All convex combinations of points in a set



It’s always the **smallest** convex superset

# Convex Sets

Important examples: Are these convex?

Hyperplane $\{x \mid a^T x = b\}$

Half-space $\{x \mid a^T x \geq b\}$

Sphere $\{x \mid \|x - x_0\| = b\}$

Ball $\{x \mid \|x - x_0\| \leq b\}$

Polynomials $\left\{ f \mid f = \sum_i a_i x^i \right\}$

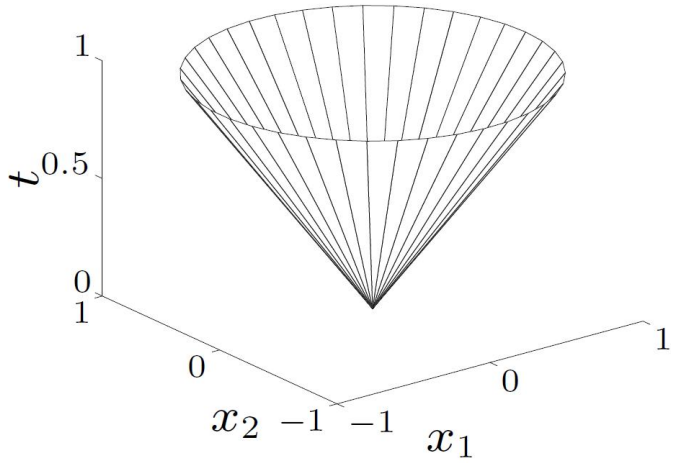
# Convex Sets

Cones (not necessarily convex)

$$x \in C \Longrightarrow ax \in C, \quad \forall a \geq 0$$

Second-order cone:

$$C_2 = \{(x, t) \mid \quad \|x\| \leq t\} \in \mathbb{R}^{n+1}$$



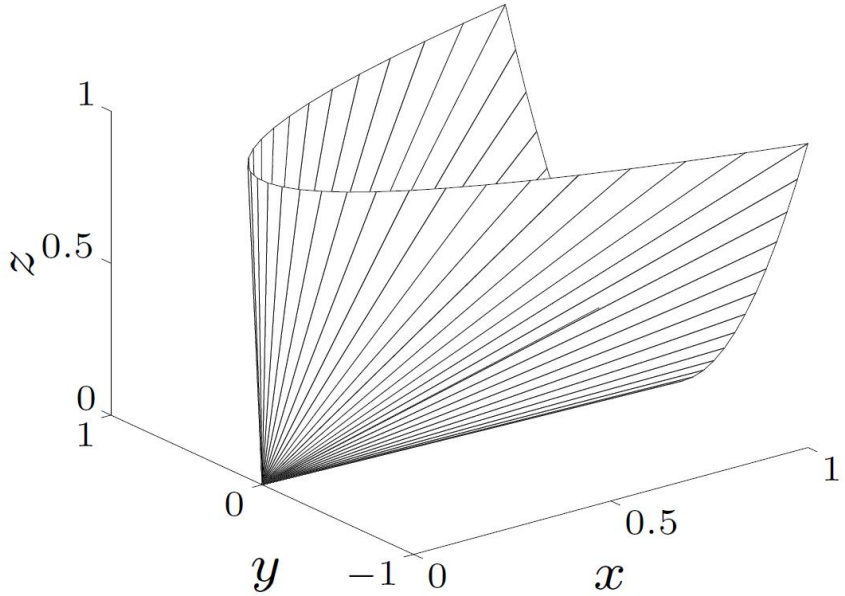
# Convex Sets

Semi-definite cone

$$\mathcal{S}_+^n = \left\{ A \in \mathbb{R}^{n \times n} \mid A = A^T, A \succeq 0 \right\}$$

Set of PSD matrices?

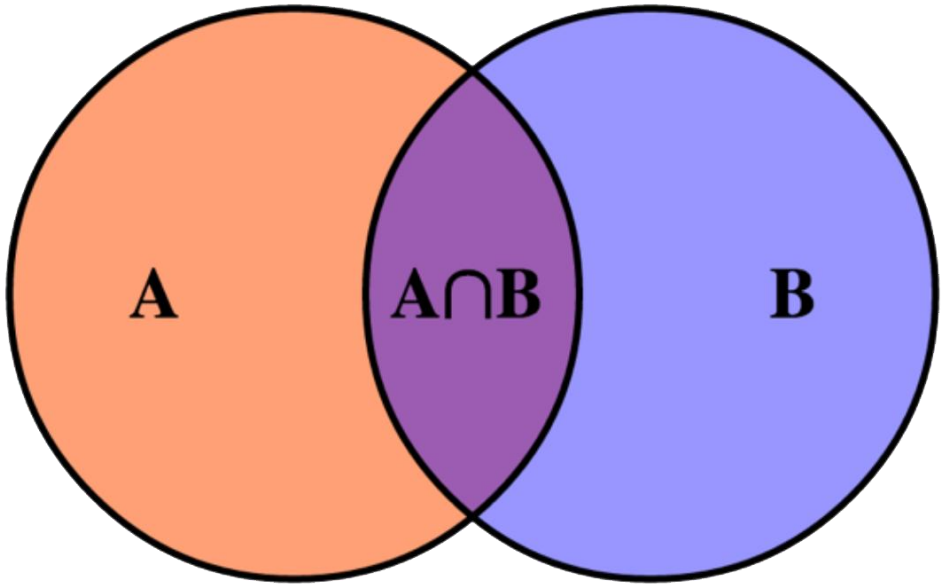
$$\begin{bmatrix} x & y \\ y & z \end{bmatrix} \in \mathcal{S}_+^2$$




Why is this a cone?

# Convex Sets

Allowed operations

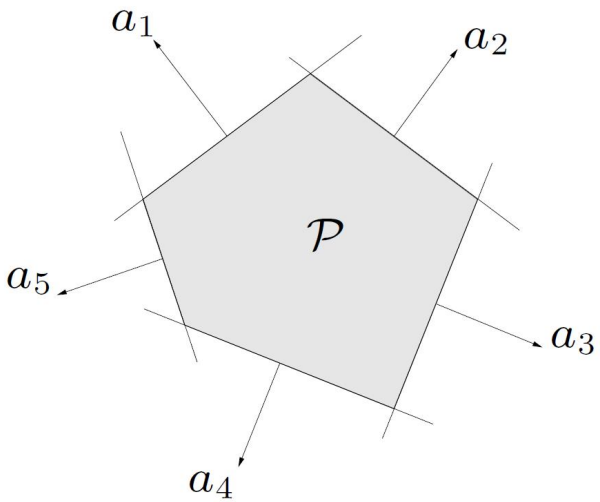


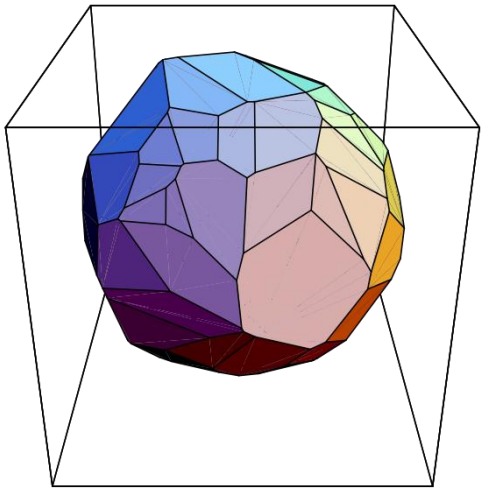

The **intersection** of convex sets is convex

# Convex Sets

Polytope

$$\{x : Ax \leq b\}$$



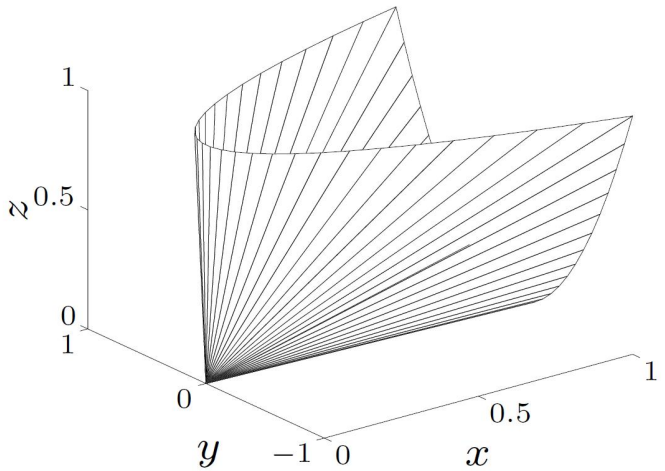
Is it convex? Why?

# Convex Sets

Semi-definite cone

$$x^T A x \geq 0, \forall x$$

$$\mathcal{S}_+^n = \bigcap_{x \in \mathbb{R}^n} \{A \mid x^T A x \geq 0\}$$




Convex?

# Convex Sets

Other allowed operations

Set sum $A + B = \{x + y \mid x \in A, y \in B\}$

Set product $A \times B = \{(x, y) \mid x \in A, y \in B\}$

What about union?

$$A \cup B$$

# High-Order Info of Functions

# High-Order Info of Functions

Function $f(x) = f(x_1, x_2, x_3)$

Gradient $\nabla f(x) = \begin{pmatrix} \partial_1 f(x) \\ \partial_2 f(x) \\ \partial_3 f(x) \end{pmatrix}$

Hessian $\nabla^2 f(x) = \begin{pmatrix} \partial_1^2 f(x) & \partial_1 \partial_2 f(x) & \partial_1 \partial_3 f(x) \\ \partial_2 \partial_1 f(x) & \partial_2^2 f(x) & \partial_2 \partial_3 f(x) \\ \partial_3 \partial_1 f(x) & \partial_3 \partial_2 f(x) & \partial_3^2 f(x) \end{pmatrix}$

Symmetric for smooth func

Approx at zero $f(x) = \underbrace{f(0) + x^T \nabla f(0)}_{\text{Linear approx}} + \underbrace{\frac{1}{2} x^T \nabla^2 f(0) x}_{\text{Quadratic approx}} + O\left(\|x - x_0\|^3\right)$

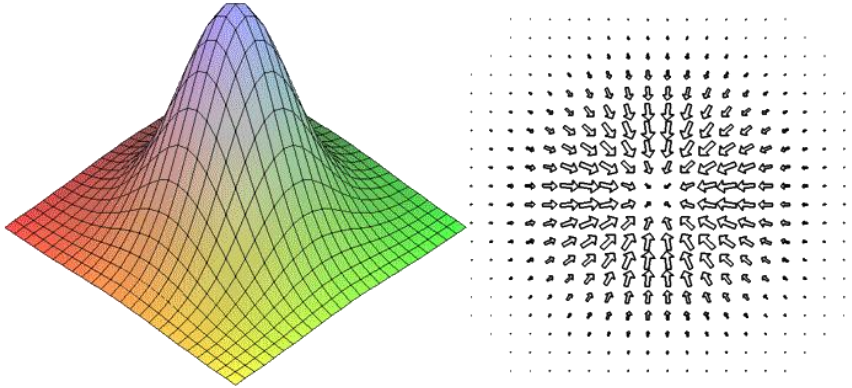
# High-Order Info of Functions

## Gradient

Let $f(x) : \mathbb{R}^n \to \mathbb{R}$, then vector, which contains all first order partial derivatives:

$$\nabla f(x) = \frac{df}{dx} = \begin{pmatrix} \frac{\partial f}{\partial x_1} \\ \frac{\partial f}{\partial x_2} \\ \vdots \\ \frac{\partial f}{\partial x_n} \end{pmatrix}$$

named gradient of $f(x)$. This vector indicates the direction of steepest ascent. Thus, vector $-\nabla f(x)$ means the direction of the steepest descent of the function in the point. Moreover, the gradient vector is always orthogonal to the contour line in the point.



# High-Order Info of Functions

## Hessian

Let $f(x) : \mathbb{R}^n \to \mathbb{R}$, then matrix, containing all the second order partial derivatives:

$$f''(x) = \frac{d(\nabla f)}{dx^T} = \frac{d\left(\nabla f^T\right)}{dx} = \begin{pmatrix} \frac{\partial^2 f}{\partial x_1 \partial x_1} & \frac{\partial^2 f}{\partial x_1 \partial x_2} & \cdots & \frac{\partial^2 f}{\partial x_1 \partial x_n} \\ \frac{\partial^2 f}{\partial x_2 \partial x_1} & \frac{\partial^2 f}{\partial x_2 \partial x_2} & \cdots & \frac{\partial^2 f}{\partial x_2 \partial x_n} \\ \vdots & \vdots & \ddots & \vdots \\ \frac{\partial^2 f}{\partial x_n \partial x_1} & \frac{\partial^2 f}{\partial x_n \partial x_2} & \cdots & \frac{\partial^2 f}{\partial x_n \partial x_n} \end{pmatrix}$$

But actually, Hessian could be a tensor in such a way: $\left(f(x) : \mathbb{R}^n \to \mathbb{R}^m\right)$ is just 3d tensor, every slice is just hessian of corresponding scalar function $\left(H\left(f_1(x)\right), H\left(f_2(x)\right), \ldots, H\left(f_m(x)\right)\right)$

## Jacobian

The extension of the gradient of multidimensional $f(x) : \mathbb{R}^n \to \mathbb{R}^m$:

$$f'(x) = \frac{df}{dx^T} = \begin{pmatrix} \frac{\partial f_1}{\partial x_1} & \frac{\partial f_1}{\partial x_2} & \cdots & \frac{\partial f_1}{\partial x_n} \\ \frac{\partial f_2}{\partial x_1} & \frac{\partial f_2}{\partial x_2} & \cdots & \frac{\partial f_2}{\partial x_n} \\ \vdots & \vdots & \ddots & \vdots \\ \frac{\partial f_m}{\partial x_1} & \frac{\partial f_m}{\partial x_2} & \cdots & \frac{\partial f_m}{\partial x_n} \end{pmatrix}$$

# High-Order Info of Functions

Jacobian of g(x):

$$\begin{pmatrix} \frac{\partial g_1}{\partial x_1} & \frac{\partial g_1}{\partial x_2} & \cdots & \frac{\partial g_1}{\partial x_n} \\ \frac{\partial g_2}{\partial x_1} & \frac{\partial g_2}{\partial x_2} & \cdots & \frac{\partial g_2}{\partial x_n} \\ \vdots & \vdots & \ddots & \vdots \\ \frac{\partial g_m}{\partial x_1} & \frac{\partial g_m}{\partial x_2} & \cdots & \frac{\partial g_m}{\partial x_n} \end{pmatrix}$$

Substitution:

$$g(x) := \nabla f(x),\ g_i(x) = \frac{\partial f}{\partial x_i}$$

The Hessian of f(x):

$$\begin{pmatrix} \frac{\partial^2 f}{\partial x_1 \partial x_1} & \frac{\partial^2 f}{\partial x_1 \partial x_2} & \cdots & \frac{\partial^2 f}{\partial x_1 \partial x_n} \\ \frac{\partial^2 f}{\partial x_2 \partial x_1} & \frac{\partial^2 f}{\partial x_2 \partial x_2} & \cdots & \frac{\partial^2 f}{\partial x_2 \partial x_n} \\ \vdots & \vdots & \ddots & \vdots \\ \frac{\partial^2 f}{\partial x_n \partial x_1} & \frac{\partial^2 f}{\partial x_n \partial x_2} & \cdots & \frac{\partial^2 f}{\partial x_n \partial x_n} \end{pmatrix}$$

# High-Order Info of Functions

## Matrix and vector derivatives

Table

$$f(x) : X \to Y; \qquad \frac{\partial f(x)}{\partial x} \in G$$

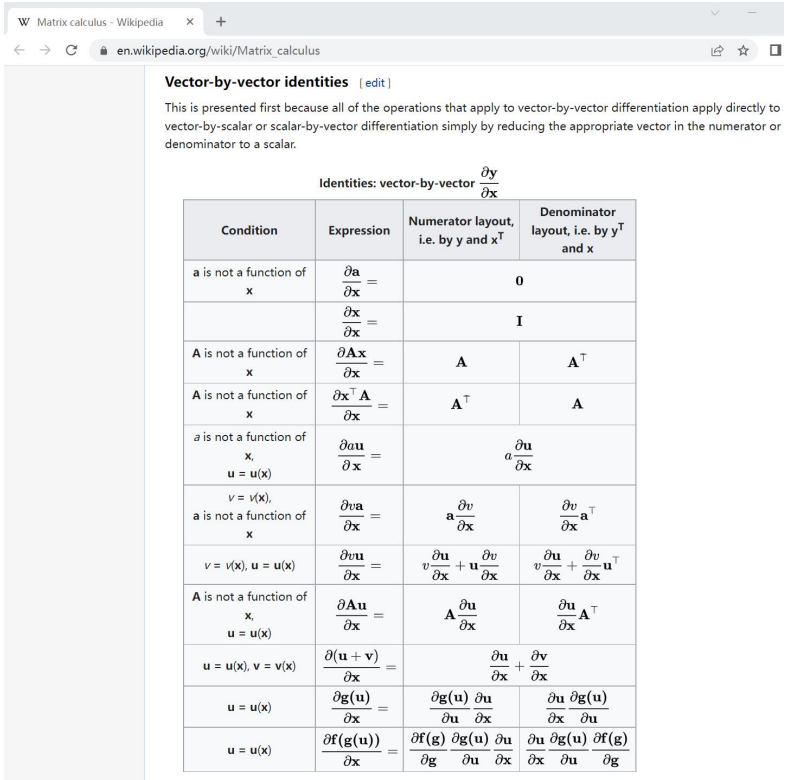
| X | Y | G | Name |
|---|---|---|---|
| $\mathbb{R}$ | $\mathbb{R}$ | $\mathbb{R}$ | $f'(x)$ (derivative) |
| $\mathbb{R}^n$ | $\mathbb{R}$ | $\mathbb{R}^n$ | $\frac{\partial f}{\partial x_i}$ (gradient) |
| $\mathbb{R}^n$ | $\mathbb{R}^m$ | $\mathbb{R}^{n\times m}$ | $\frac{\partial f_i}{\partial x_j}$ (jacobian) |
| $\mathbb{R}^{m\times n}$ | $\mathbb{R}$ | $\mathbb{R}^{m\times n}$ | $\frac{\partial f}{\partial x_{ij}}$ |

## The notation of differentials could be useful here

$$dA = 0$$
$$d(\alpha X) = \alpha(dX)$$
$$d(AXB) = A(dX)B$$
$$d(X + Y) = dX + dY$$
$$d\left(X^\top\right) = (dX)^\top$$
$$d(XY) = (dX)Y + X(dY)$$
$$d\langle X, Y\rangle = \langle dX, Y\rangle + \langle X, dY\rangle$$
$$d\left(\frac{X}{\phi}\right) = \frac{\phi dX - (d\phi)X}{\phi^2}$$
$$d\,\mathrm{tr}\,X = I$$
$$df(g(x)) = \frac{df}{dg} \cdot dg(x)$$

# High-Order Info of Functions

https://en.wikipedia.org/wiki/Matrix_calculus




**Vector-by-vector identities** [ edit ]

This is presented first because all of the operations that apply to vector-by-vector differentiation apply directly to vector-by-scalar or scalar-by-vector differentiation simply by reducing the appropriate vector in the numerator or denominator to a scalar.

**Identities: vector-by-vector $\frac{\partial \mathbf{y}}{\partial \mathbf{x}}$**

| Condition | Expression | Numerator layout, i.e. by y and $\mathbf{x}^\mathsf{T}$ | Denominator layout, i.e. by $\mathbf{y}^\mathsf{T}$ and x |
|---|---|---|---|
| **a** is not a function of **x** | $\frac{\partial \mathbf{a}}{\partial \mathbf{x}} =$ | $\mathbf{0}$ | |
| | $\frac{\partial \mathbf{x}}{\partial \mathbf{x}} =$ | $\mathbf{I}$ | |
| **A** is not a function of **x** | $\frac{\partial \mathbf{A}\mathbf{x}}{\partial \mathbf{x}} =$ | $\mathbf{A}$ | $\mathbf{A}^\mathsf{T}$ |
| **A** is not a function of **x** | $\frac{\partial \mathbf{x}^\mathsf{T}\mathbf{A}}{\partial \mathbf{x}} =$ | $\mathbf{A}^\mathsf{T}$ | $\mathbf{A}$ |
| *a* is not a function of **x**, **u** = **u**(**x**) | $\frac{\partial a\mathbf{u}}{\partial \mathbf{x}} =$ | $a\frac{\partial \mathbf{u}}{\partial \mathbf{x}}$ | |
| *v* = *v*(**x**), **a** is not a function of **x** | $\frac{\partial v\mathbf{a}}{\partial \mathbf{x}} =$ | $\mathbf{a}\frac{\partial v}{\partial \mathbf{x}}$ | $\frac{\partial v}{\partial \mathbf{x}}\mathbf{a}^\mathsf{T}$ |
| *v* = *v*(**x**), **u** = **u**(**x**) | $\frac{\partial v\mathbf{u}}{\partial \mathbf{x}} =$ | $v\frac{\partial \mathbf{u}}{\partial \mathbf{x}} + \mathbf{u}\frac{\partial v}{\partial \mathbf{x}}$ | $v\frac{\partial \mathbf{u}}{\partial \mathbf{x}} + \frac{\partial v}{\partial \mathbf{x}}\mathbf{u}^\mathsf{T}$ |
| **A** is not a function of **x**, **u** = **u**(**x**) | $\frac{\partial \mathbf{A}\mathbf{u}}{\partial \mathbf{x}} =$ | $\mathbf{A}\frac{\partial \mathbf{u}}{\partial \mathbf{x}}$ | $\frac{\partial \mathbf{u}}{\partial \mathbf{x}}\mathbf{A}^\mathsf{T}$ |
| **u** = **u**(**x**), **v** = **v**(**x**) | $\frac{\partial (\mathbf{u}+\mathbf{v})}{\partial \mathbf{x}} =$ | $\frac{\partial \mathbf{u}}{\partial \mathbf{x}} + \frac{\partial \mathbf{v}}{\partial \mathbf{x}}$ | |
| **u** = **u**(**x**) | $\frac{\partial \mathbf{g}(\mathbf{u})}{\partial \mathbf{x}} =$ | $\frac{\partial \mathbf{g}(\mathbf{u})}{\partial \mathbf{u}}\frac{\partial \mathbf{u}}{\partial \mathbf{x}}$ | $\frac{\partial \mathbf{u}}{\partial \mathbf{x}}\frac{\partial \mathbf{g}(\mathbf{u})}{\partial \mathbf{u}}$ |
| **u** = **u**(**x**) | $\frac{\partial \mathbf{f}(\mathbf{g}(\mathbf{u}))}{\partial \mathbf{x}} =$ | $\frac{\partial \mathbf{f}(\mathbf{g})}{\partial \mathbf{g}}\frac{\partial \mathbf{g}(\mathbf{u})}{\partial \mathbf{u}}\frac{\partial \mathbf{u}}{\partial \mathbf{x}}$ | $\frac{\partial \mathbf{u}}{\partial \mathbf{x}}\frac{\partial \mathbf{g}(\mathbf{u})}{\partial \mathbf{u}}\frac{\partial \mathbf{f}(\mathbf{g})}{\partial \mathbf{g}}$ |

# High-Order Info of Functions

Examples

Calculate $\nabla f(x)$, if $f(x) = \frac{1}{2}x^T Ax + b^T x + c$

$$f(x) = \frac{1}{2}\sum_{i,j=1}^{n}[x_i a_{ij} x_j] + \sum_{i=1}^{n} b_i x_i + c_i$$

$$\frac{\partial f(x)}{\partial x_i} = \frac{1}{2}\sum_{i,j=1}^{n}(a_{ij} + a_{ji})x_i + b_i \rightarrow \nabla f(x) = \frac{1}{2}\left(A + A^T\right)x + b$$

Calculate $\nabla f(X)$, if $f(X) = \operatorname{tr} AX$

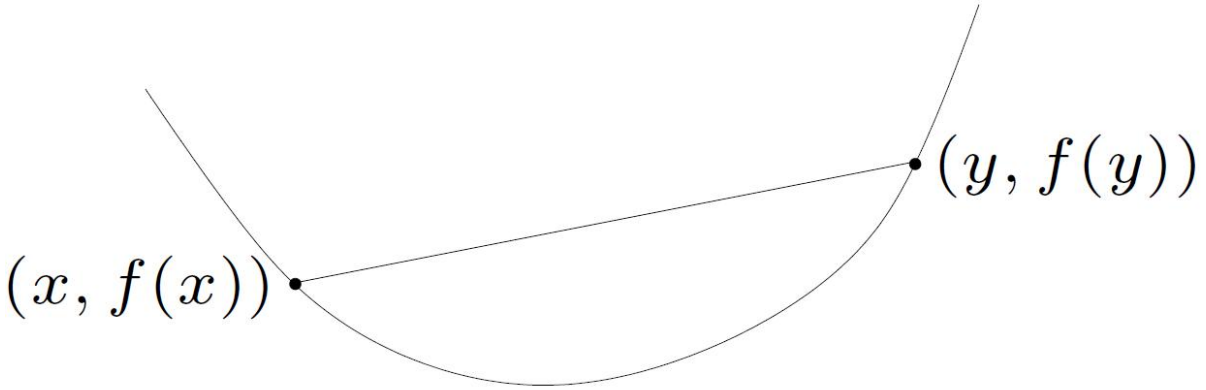
$$g(Y) = \operatorname{tr} Y$$
$$h(X) = AX$$
$$\frac{df}{dX} = \frac{dg(h(X))}{dX} = \left(\frac{dh}{dX}\right)^{\mathrm{T}} \frac{dg}{dh} = (A)^{\mathrm{T}} I = A^{\mathrm{T}}$$

# Convex Functions

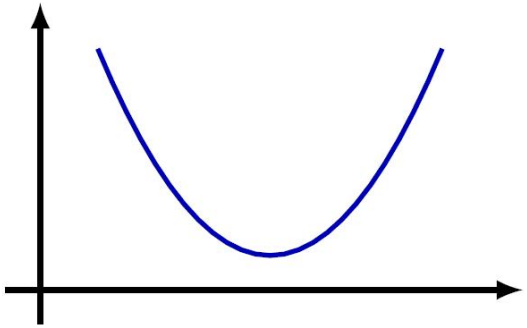
# Convex Functions

Jensen's inequality

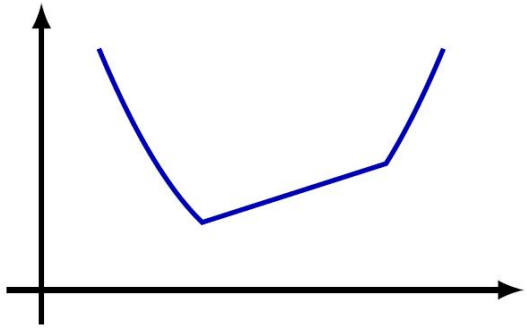
$$f(\theta x + (1-\theta)y) \leq \theta f(x) + (1-\theta)f(y)$$



# Convex Functions

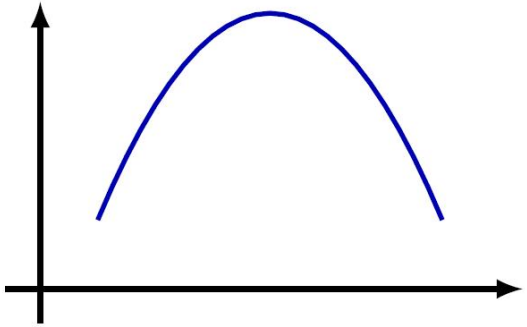


strictly convex function

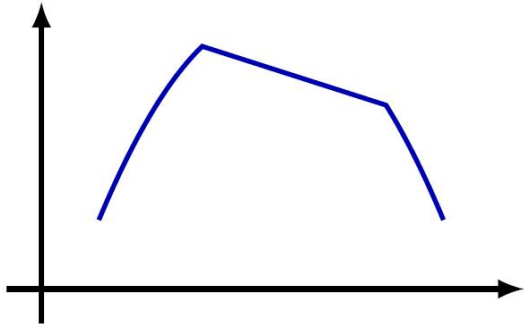


convex function

$$f(\theta x + (1-\theta)y) \leq \theta f(x) + (1-\theta) f(y)$$



strictly concave function

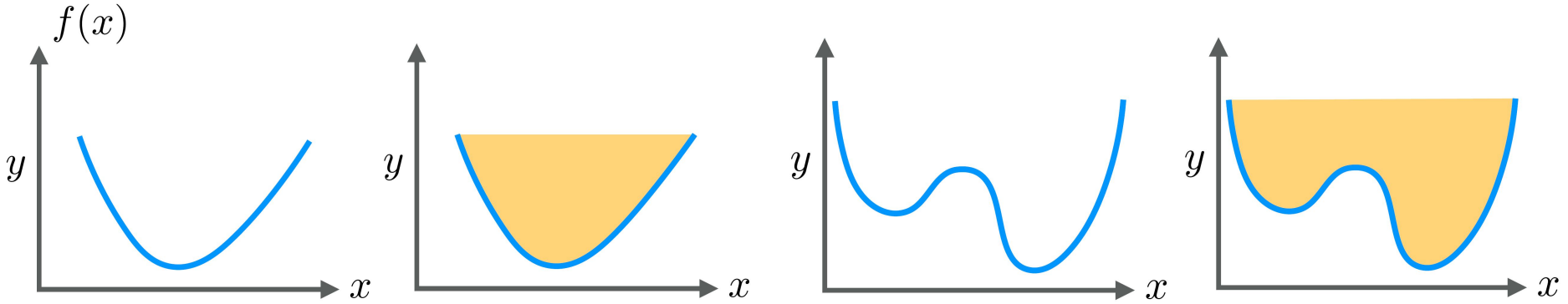


concave function

# Convex Functions

Epigraph (上方图)

$$\text{epi}(f) = \{(x, y) \mid f(x) \leq y\}$$

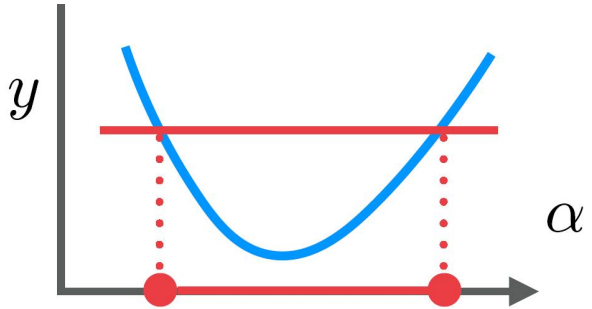



convex function = convex epigraph

# Convex Functions

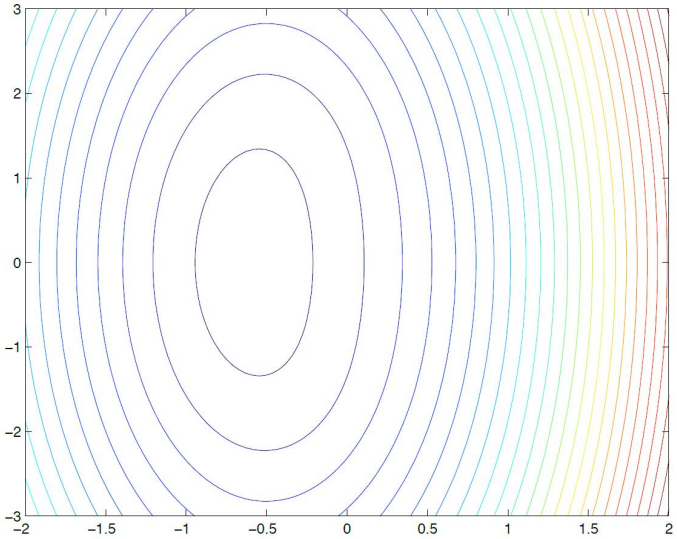
Why do we care about convex functions?

Convex functions have **convex sub-level sets**




sub-level set

$$f_\alpha = \{x \mid f(x) \leq \alpha\}$$

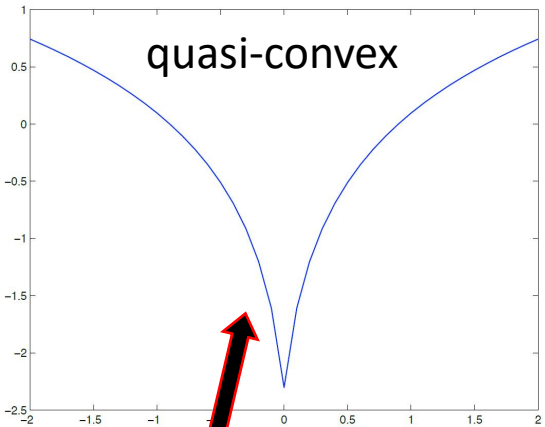

convex contours

# Convex Functions

Why do we care about convex functions?

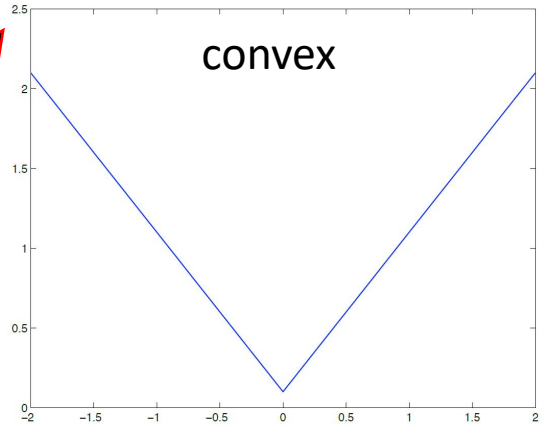
Convexity are somewhat easy to **preserve**

$f(x) = \log(|x| + .1)$




$e^f$

$f(x) = |x| + .1$




convex contours = quasi-convex > convex

Sum of quasi-convex functions is not necessary quasi-convex.

# Convex Functions

Why do we care about convex functions?

Any minimizer is **global** (why)

Set of minima is **convex** (why)

Convex functions are **closed under many operations**.

# Convex Functions

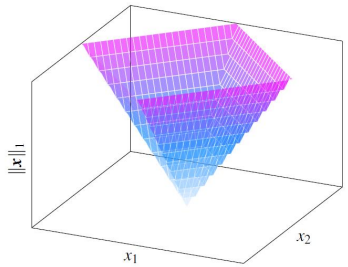
Positive weighted sum

$$g(x) = \sum_i f_i(x)$$

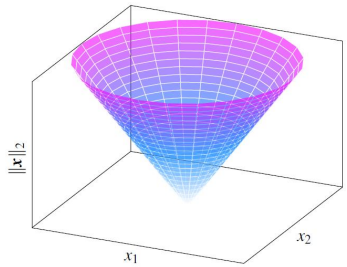
Examples:

$$f(x) = \|x\|^2 = \sum_i x_i^2$$

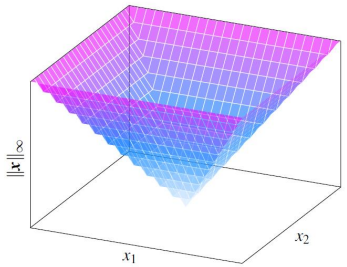
$$f(x) = x + x^2 + x^6 + |x|$$




1-norm




2-norm




$\infty$-norm

Sums of convex functions are convex

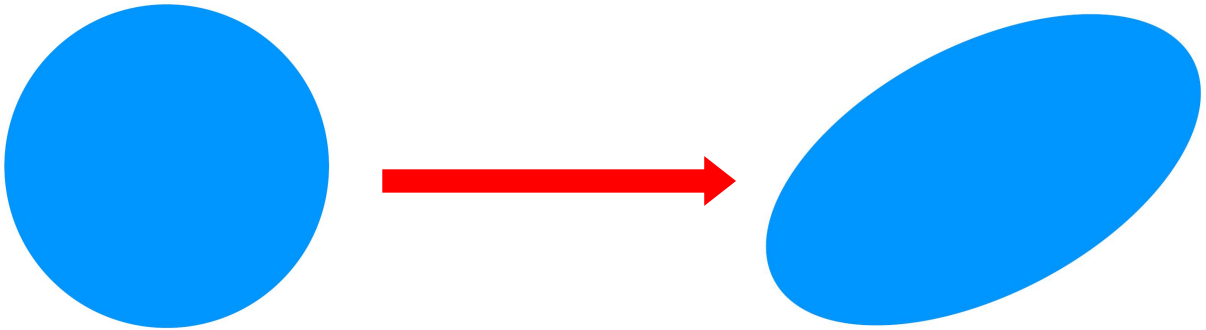
# Convex Functions

Affine composition

Affine composition with convex function = convex

$$f(x) \longrightarrow f(Ax + b)$$

Is the convexity of epigraphs preserved by these operations?



# Convex Functions

Point-wise max preserves convexity

$$g(x) = \max_i f_i(x)$$

Absolute value $|x| = \max\{x, -x\}$

Infinity norm $\|x\|_\infty = \max_i |x_i|$

Max eigenvalue $\|A\|_2 = \max_v v^T A v$

Is the convexity of epigraphs preserved by these operations?

# Convex Functions

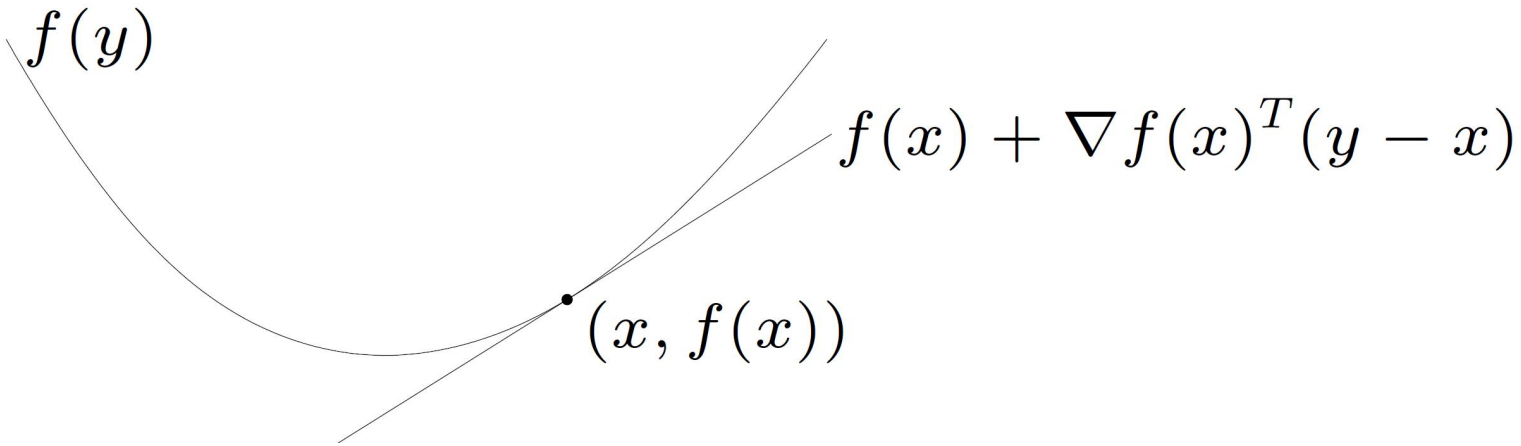
Why are these convex?

| | | |
|---|---|---|
| Trace | $f(X) = \text{trace}(A^T X)$ | Linear operator |
| Distance over set | $f(x) = \max_{y \in C} \|x - y\|$ | Max over convex |
| Distance to convex set | $f(x) = \min_{y \in C} \|x - y\|$ | Special case |
| Affine Norm | $f(x) = \|b + \sum_i A_i x_i\|_2$ | Affine comp |

If g(x,y) is convex, then minimizing for y preserves convexity

# Convex Function Property

Convex functions lie **ABOVE** their linear approximation




$$f(y) \geq f(x) + \nabla f(x)^T(y - x) \quad \text{for all } x, y \in \text{dom } f$$

# Convex Function Property

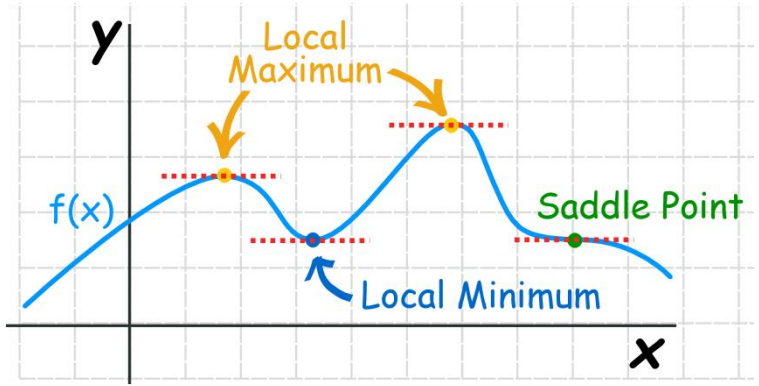
First-order conditions = optimality for convex funcs only

$$f(y) \geq f(x) + \nabla f(x)^T (y - x)$$

$$= 0$$

$$f(y) \geq f(x)$$

This doesn't happen!



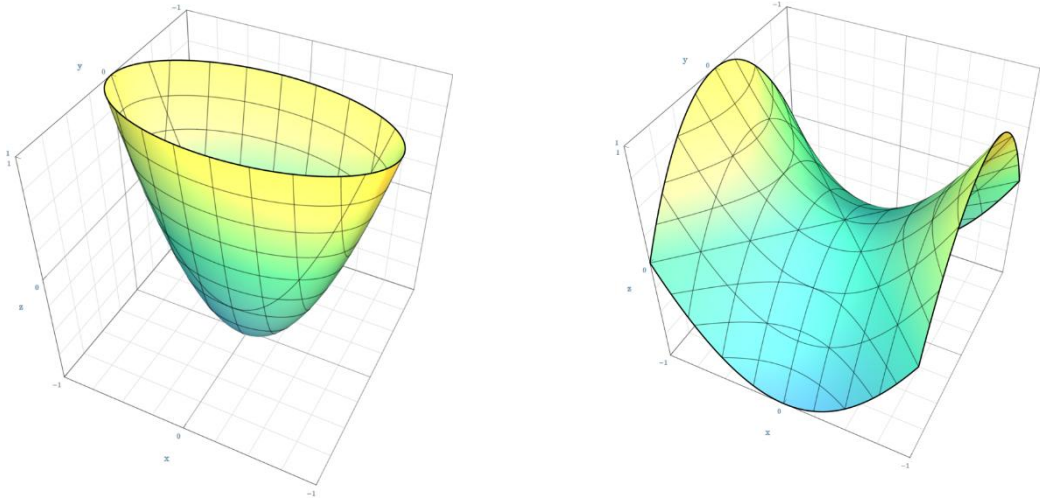

**Any** minima is a **global** minima

# Convex Function Property

Second-order conditions

A smooth function is convex iff $\nabla^2 f(x) \succeq 0, \quad \forall x$ (semi-definite)

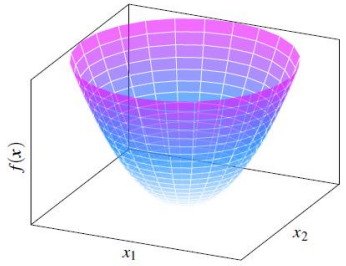
For non-convex functions, minima satisfy: $\nabla^2 f(x^\star) \succeq 0$



The Hessian is a good local model of a smooth function

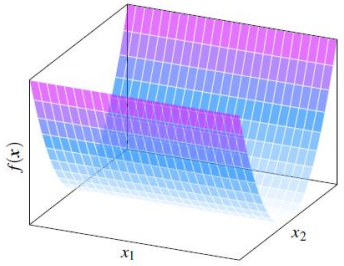
# Convex Function Property

Quadratic functions $f(x) = x^{\mathrm{T}} Q x \in \mathbb{R}^2$



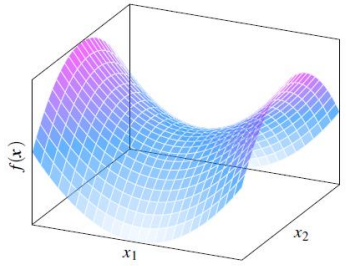

$Q = \mathrm{diag}\{1, 1\}$

strictly convex



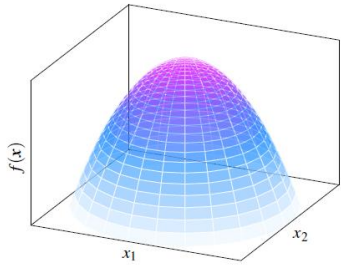

$Q = \mathrm{diag}\{0, 1\}$

convex



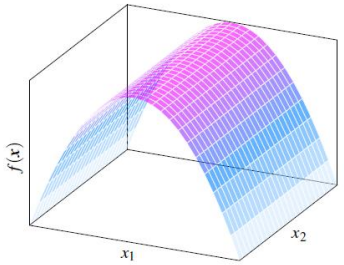

$Q = \mathrm{diag}\{1, -1\}$

neither convex nor concave




$Q = \mathrm{diag}\{-1, -1\}$

strictly concave




$Q = \mathrm{diag}\{-1, 0\}$

concave

# Convex Function Property

Strong convexity

$$f(y) \geq f(x) + (y-x)^T \nabla f(x) + \frac{m}{2}\|y-x\|^2$$

holds for any convex func

min curvature

When Hessian exists

$$f(y) \approx f(x) + (y-x)^T \nabla f(x) + \frac{1}{2}(y-x)^T \nabla^2 f(x)(y-x)$$

$$\geq f(x) + (y-x)^T \nabla f(x) + \frac{\lambda_{\min}}{2}\|y-x\|^2$$

and so

$$\nabla^2 f(x) \succeq mI$$

# Convex Function Property

The Lipschitz constant $M$ of $\nabla f(x)$ satisfies

$$\|\nabla f(x) - \nabla f(y)\| \leq M\|y - x\|$$

$$f(y) \leq f(x) + (y - x)^T \nabla f(x) + \frac{M}{2}\|y - x\|^2$$

We can bound the objective error in terms of distance from minimizer

$$f(y) - f(x^\star) \geq \frac{m}{2}\|y - x^\star\|^2 \qquad f(y) - f(x^\star) \leq \frac{M}{2}\|y - x^\star\|^2$$

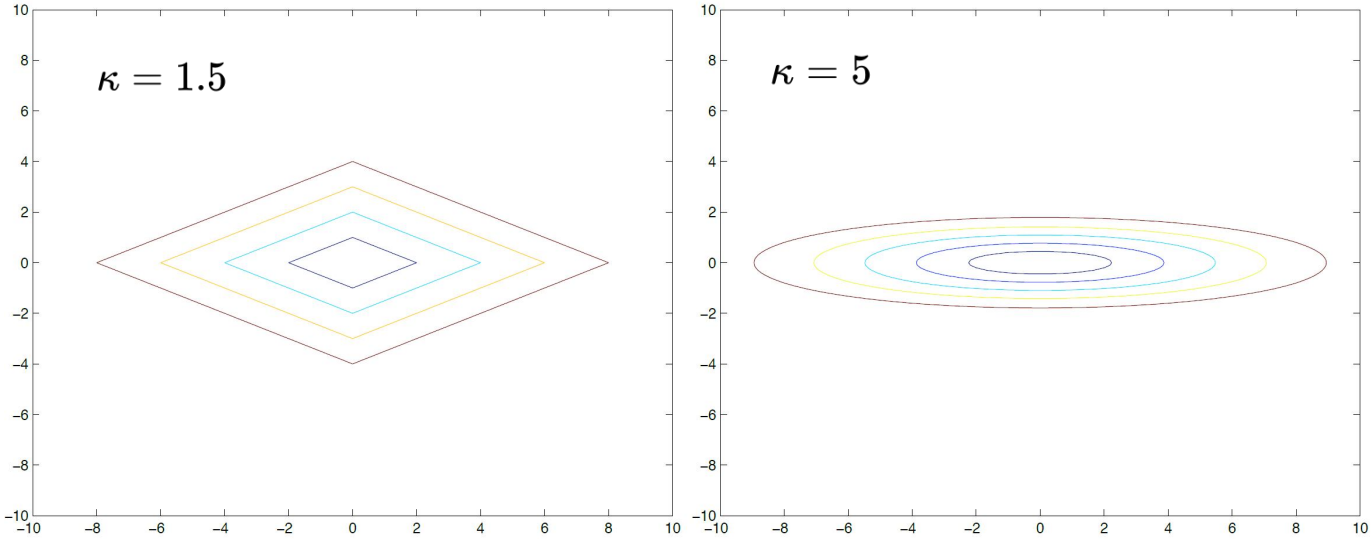
# Convex Function Property

Condition number

For any function $\kappa = \dfrac{\text{major axis}}{\text{minor axis}}$

For smooth functions $\kappa \approx \text{cond}\left(\nabla^2 f(x)\right)$

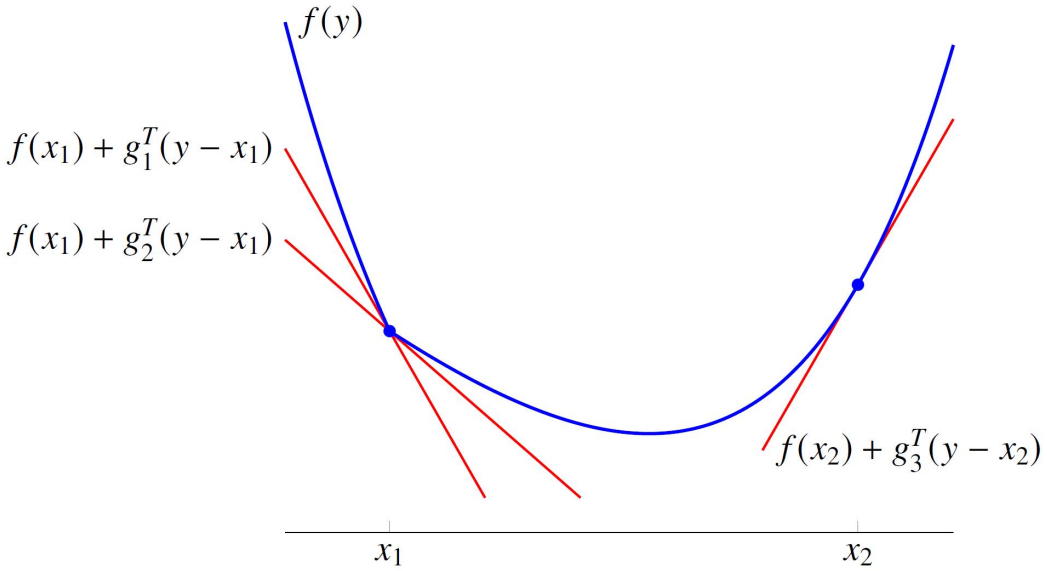
For differentiable functions $\kappa = M/m$



# Convex Function Property

Sub-differential

$$\partial f(x) = \{g : f(y) > f(x) + (y - x)^T g, \forall y\}$$

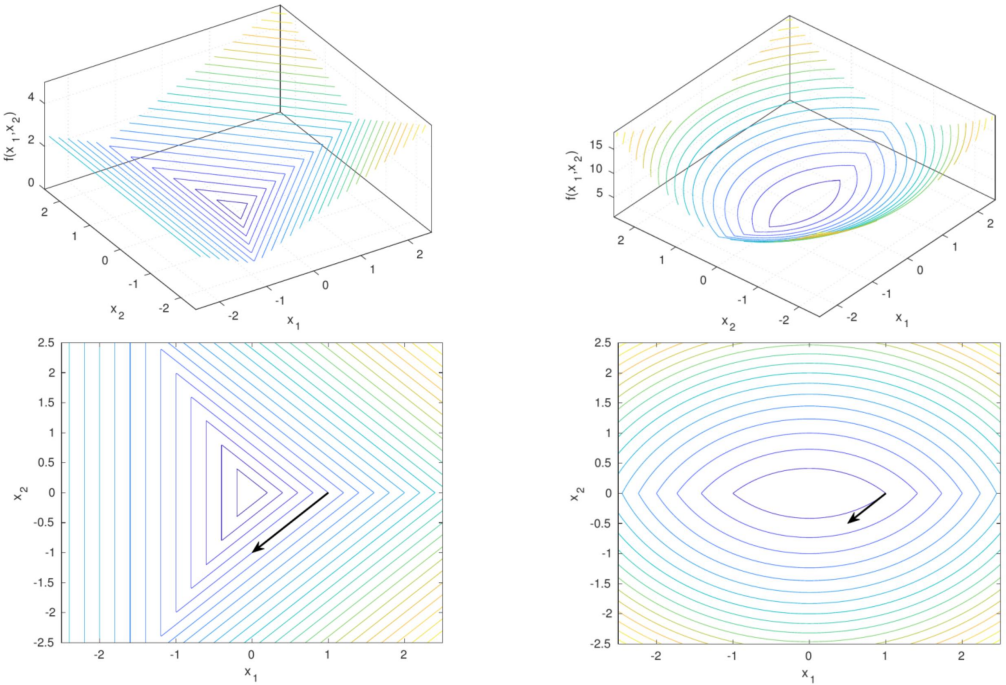



Optimality: $0 \in \partial f(x^*)$

# Convex Function Property

A direction in negative sub-differential is not necessary a descent one

$f(\boldsymbol{x}) = \max[-x_1, x_1 - x_2, x_1 + x_2]$ $f(\boldsymbol{x}) = \max\left[x_1^2 + (x_2+1)^2, x_1^2 + (x_2-1)^2\right]$

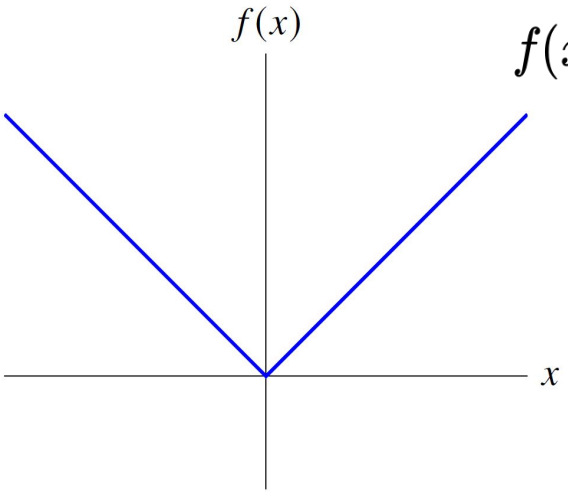



Steepest descent direction is the least-norm one in neg sub-diff.

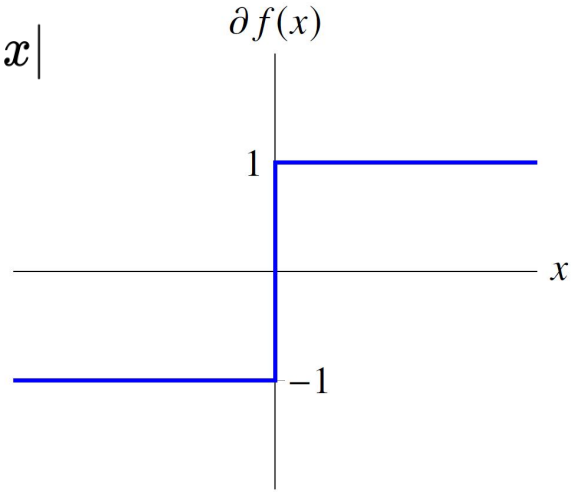
# Convex Function Property

Sub-differential example

$$\partial f(x) = \{g : f(y) > f(x) + (y - x)^T g, \forall y\}$$




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




$$\partial f(0) = [-1, 1]$$

# Convex Function Property

Monotonicity: The (sub) gradient of any convex func is monotone

$$\langle y - x, \nabla f(y) - \nabla f(x) \rangle \geq 0$$

or

$$\langle y - x, g_y - g_x \rangle \geq 0,\ g_x \in \partial f(x),\ g_y \in \partial f(y)$$

This can be obtained by adding two equations below.

$$f(y) \geq f(x) + \nabla f(x)^T (y - x)$$

$$f(x) \geq f(y) + \nabla f(y)^T (x - y)$$

# Unconstrained Optimization for Nonconvex Functions

# Unconstrained Optimization

unconstrained optimization

$$\min f(x)$$

$x = (x_1, \cdots, x_n) \in \mathbb{R}^n$: optimization variables

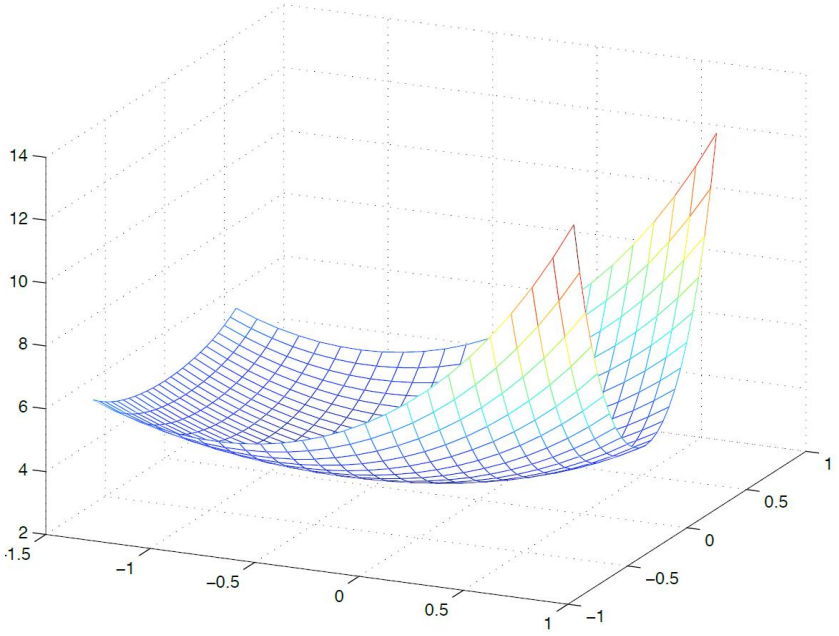
$f$: $\mathbb{R}^n \mapsto \mathbb{R}$ : objective function

**optimal solution** $x^*$ has smallest value $f$ among all vectors.

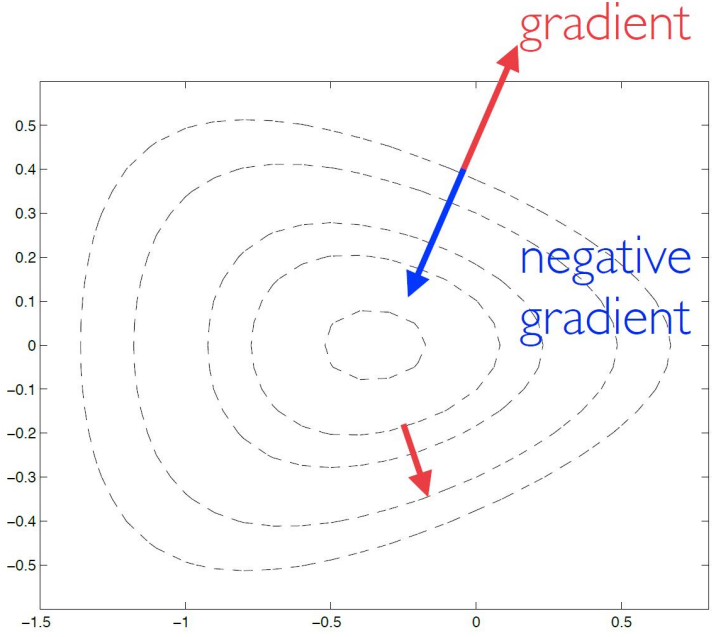
# Line-Search Steepest Gradient Descent

# Steepest Gradient Descent

Convex function




Iso-contours



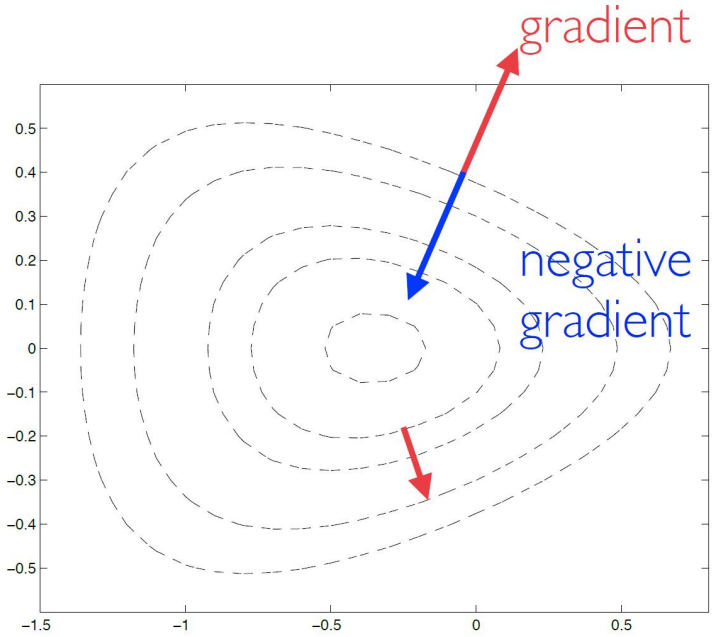
# Steepest Gradient Descent

Steepest gradient descent

$$x^{k+1} = x^k - \tau \nabla f(x^k)$$

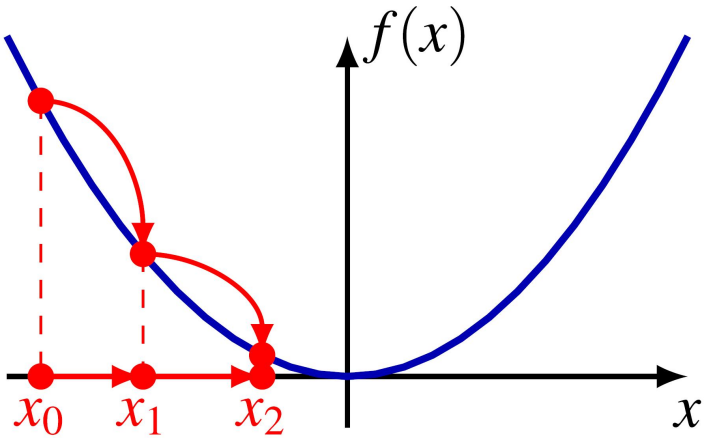
Grad or Least-norm sub-grad

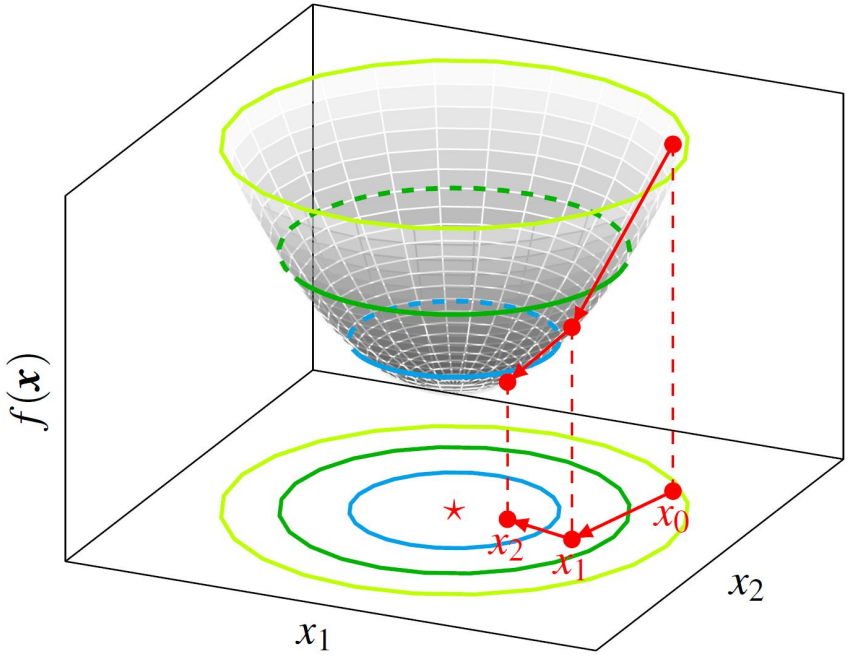
Iso-contours



# Steepest Gradient Descent

Iteration: $x^{k+1} = x^k - \tau \nabla f(x^k)$

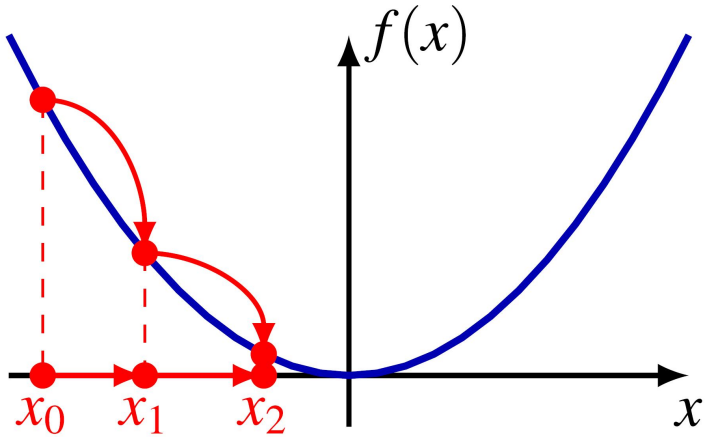



How to choose step size?

# Steepest Gradient Descent

$$x^{k+1} = x^k + \tau d,$$
$$d = -\nabla f(x^k)$$

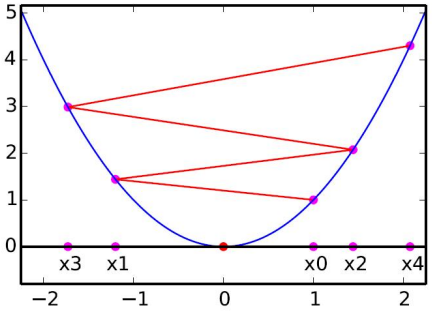


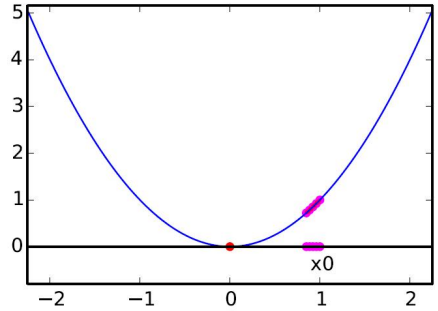

- Constant step size $\tau = c$
- Diminishing step size $\tau = c/k$
- Exact line search $\tau = \arg\min_{\alpha} f(x^k + \alpha d)$
- Inexact line search $\tau \in \{\alpha \mid f(x^k) - f(x^k + \alpha d) \geq -c \cdot \alpha d^{\mathrm{T}} \nabla f(x^k)\}$

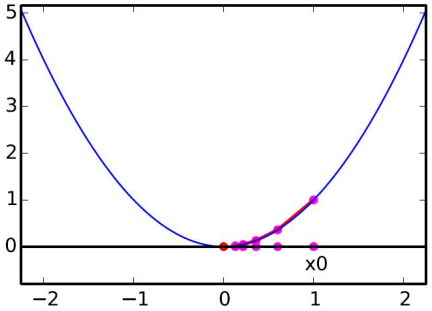
# Steepest Gradient Descent

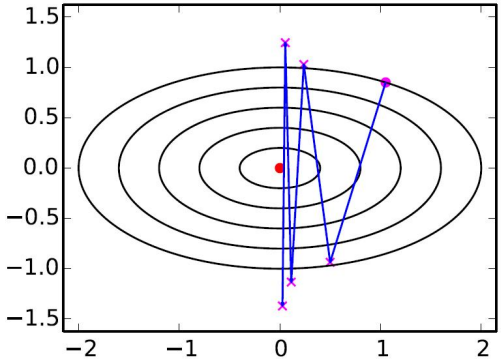
Constant step size

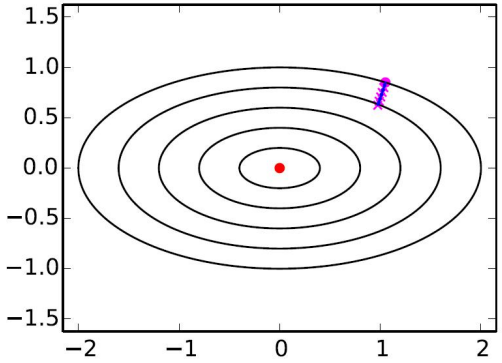
- Too large: may oscillate and diverge
- Too small: may be too slow
- “Just right”: fast convergence

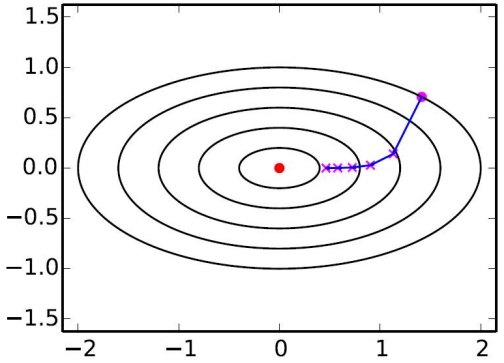






# Steepest Gradient Descent

- Constant step size $\tau = c$ Not intelligent
- Diminishing step size $\tau = c/k$ Robbins-Monro rule for expensive func
- Exact line search $\tau = \arg\min_{\alpha} f\left(x^k + \alpha d\right)$ Generally nontrivial
- Inexact line search $\tau \in \left\{\alpha \mid f(x^k) - f(x^k + \alpha d) \geq -c \cdot \alpha d^{\mathrm{T}} \nabla f(x^k)\right\}$

Armijo condition, easy to satisfy

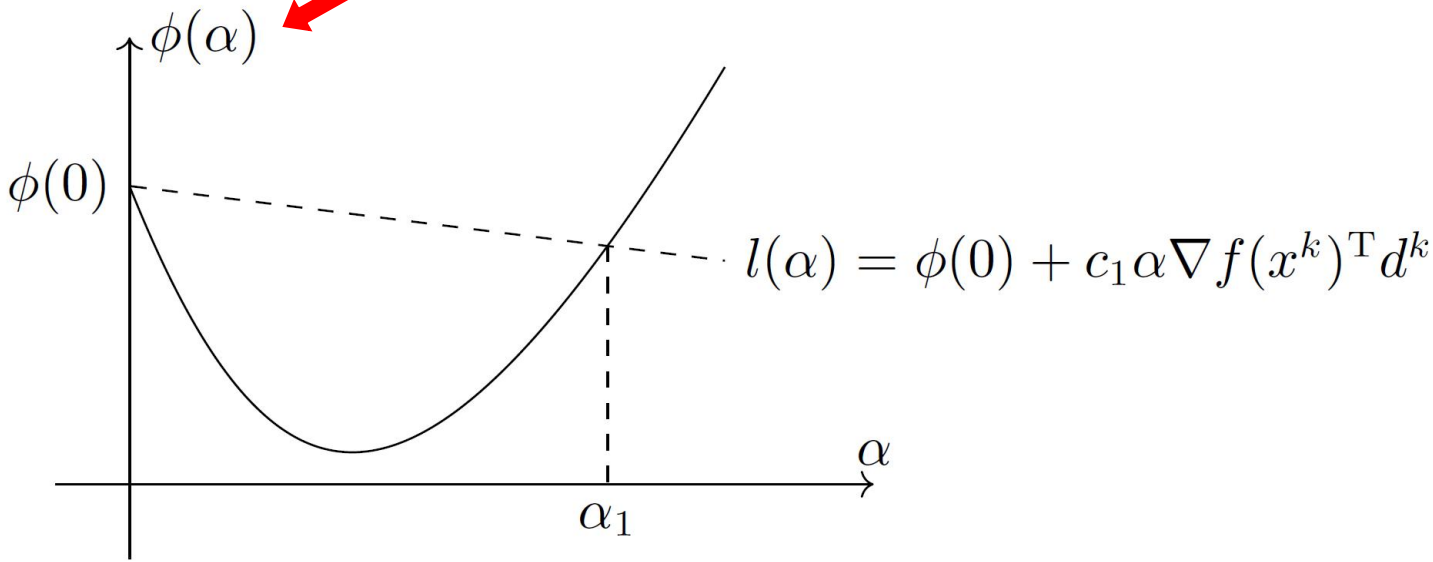
# Steepest Gradient Descent

Armijo condition (sufficient decrease condition)

$$\tau \in \left\{ \alpha \mid f(x^k) - f(x^k + \alpha d) \geq -c \cdot \alpha d^{\mathrm{T}} \nabla f(x^k) \right\}$$

$$c \in (0, 1)$$

Convex



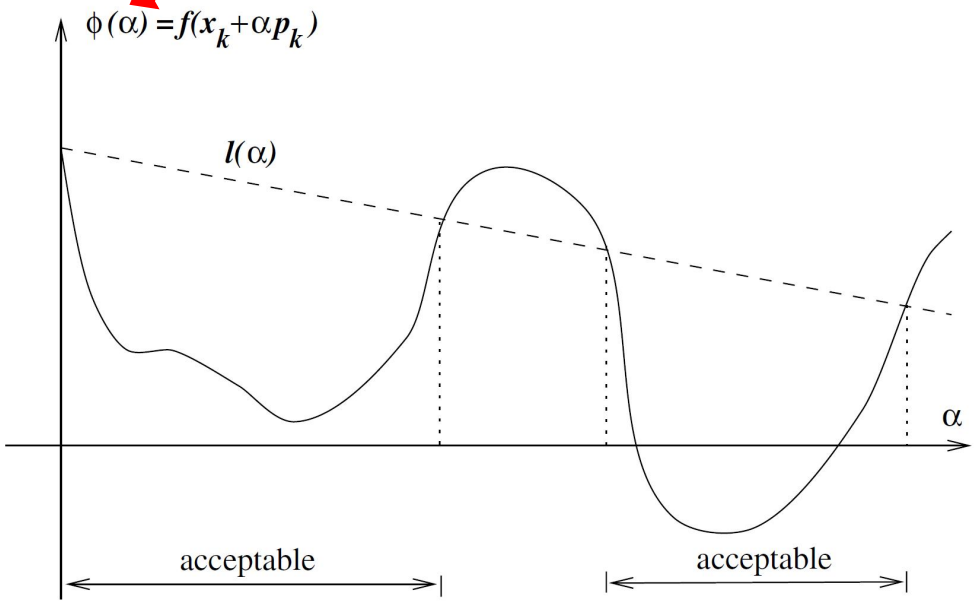
# Steepest Gradient Descent

Armijo condition (sufficient decrease condition)

$$\tau \in \left\{ \alpha \mid f(x^k) - f(x^k + \alpha d) \geq -c \cdot \alpha d^{\mathrm{T}} \nabla f(x^k) \right\}$$

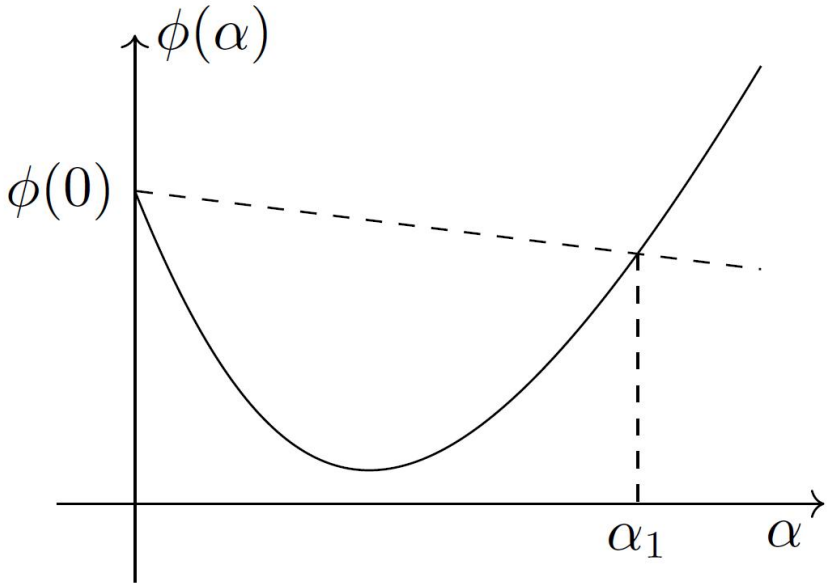
$$c \in (0, 1)$$

Nonconvex



# Steepest Gradient Descent

Backtracking/Armijo line search




Choose search direction: $d = -\nabla f(x^k)$

While $f(x^k + \tau d) > f(x^k) + c \cdot \tau d^T \nabla f(x^k)$

$\tau \leftarrow \tau/2$

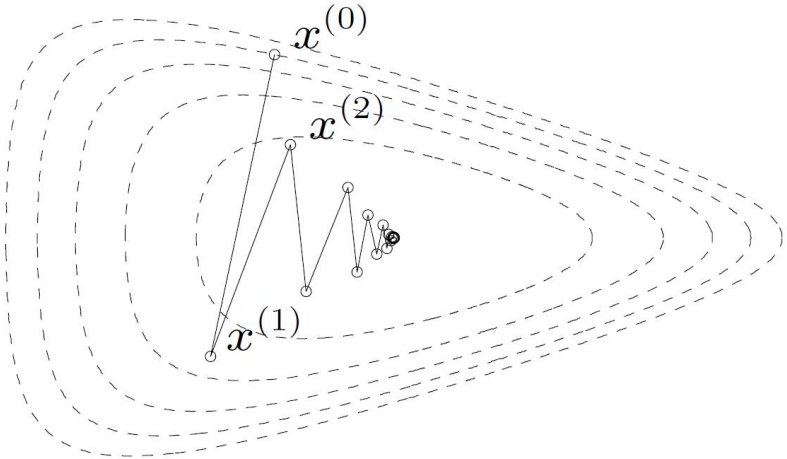
Update iterate $x^{k+1} = x^k + \tau d$

Repeat this until gradient is small or subdifferential contains zero.

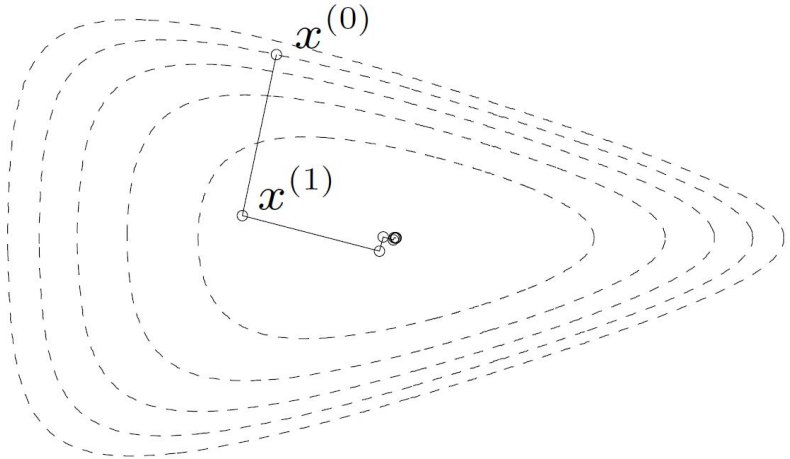
# Steepest Gradient Descent

Exact/Inexact line search

$$f(x_1, x_2) = e^{x_1+3x_2-0.1} + e^{x_1-3x_2-0.1} + e^{-x_1-0.1}$$




Inexact



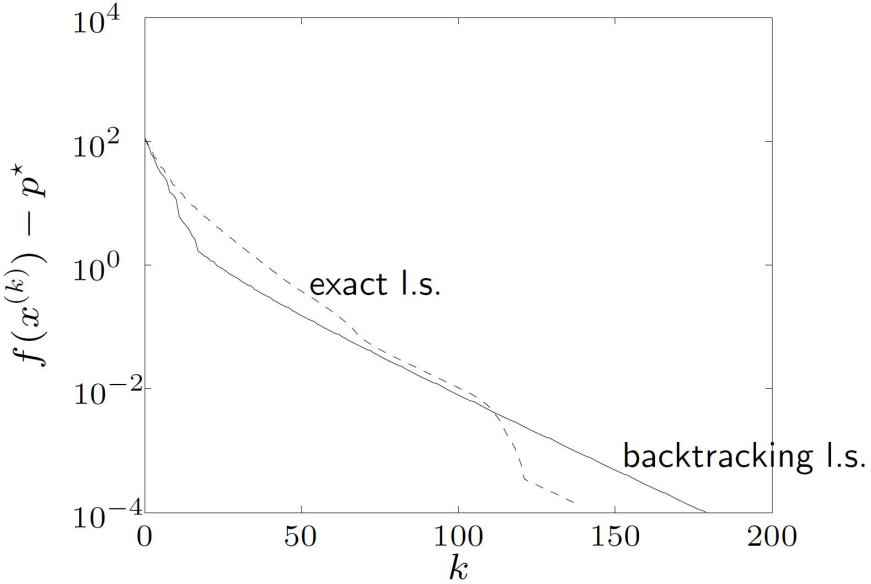

Exact

Fewer iterations do not guarantee higher efficiency

# Steepest Gradient Descent

Exact/Inexact line search

$$f(x) = c^T x - \sum_{i=1}^{500} \log\left(b_i - a_i^T x\right),\ x \in \mathbb{R}^{100}$$

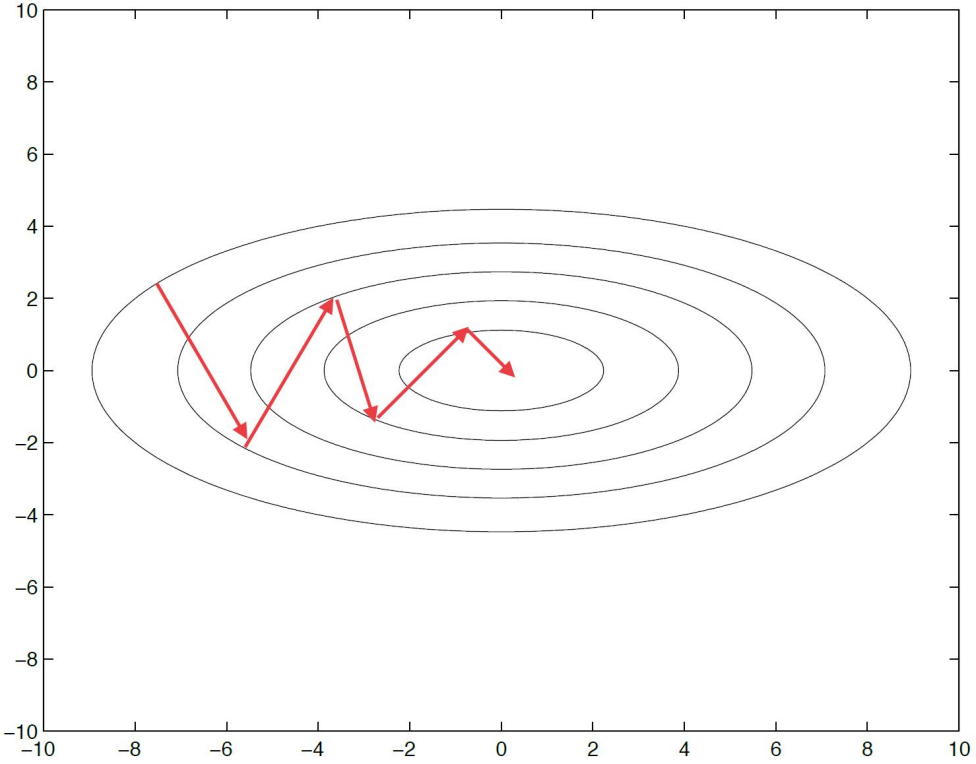



Fewer iterations do not guarantee higher efficiency

# Steepest Gradient Descent

Drawbacks: Gradients are perpendicular to contours

$$\kappa = 2$$



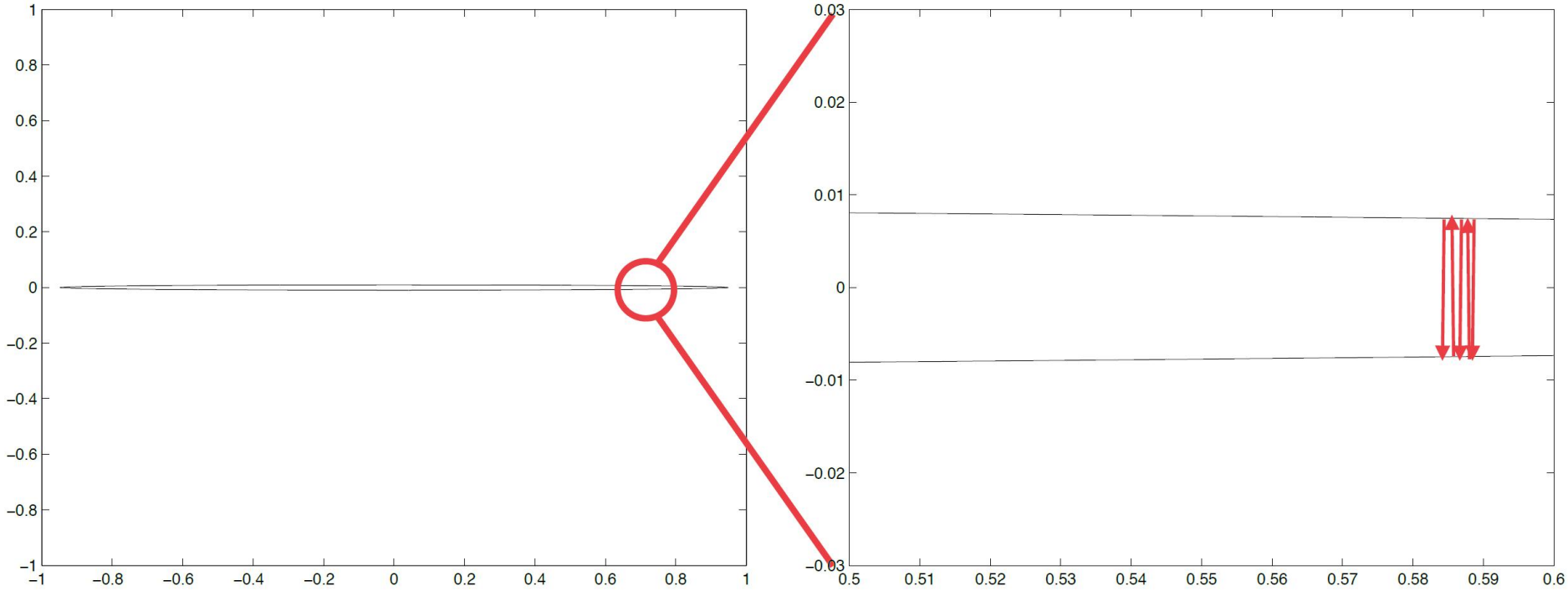
# Steepest Gradient Descent

Curvature info is needed!

Drawbacks: Poor conditioning causes performance degeneration

$\kappa = 100$

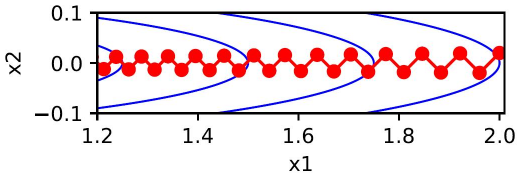
Contours are (almost) parallel!

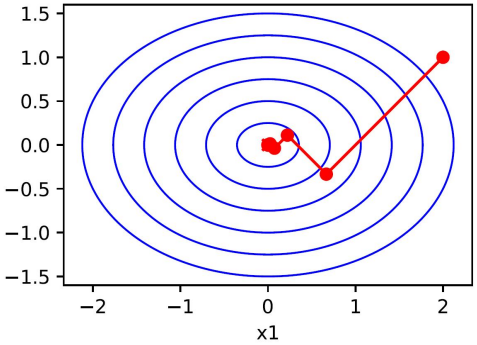


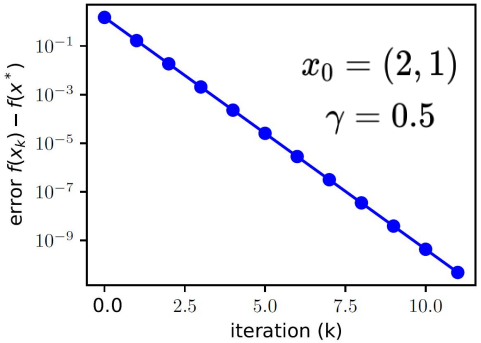
# Steepest Gradient Descent

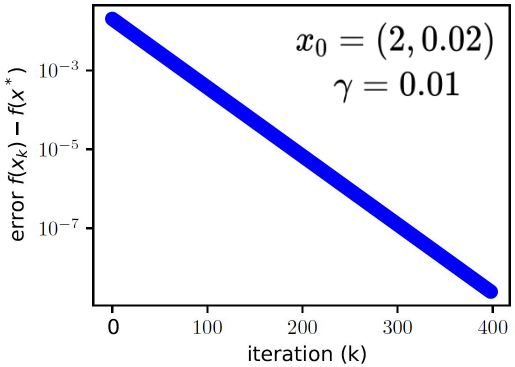
Drawbacks: Poor conditioning causes performance degeneration

$$f(x_1, x_2) = \frac{1}{2}\boldsymbol{x}^T\boldsymbol{Q}\boldsymbol{x} = \frac{\gamma}{2}x_1^2 + \frac{1}{2}x_2^2, \quad \boldsymbol{Q} = \mathrm{diag}\{\gamma, 1\}$$




Curvature info is needed!

# Modified Damped Newton's Method

# Newton's Method

By second-order Taylor expansion,

$$f(\boldsymbol{x}) \approx \hat{f}(\boldsymbol{x}) \triangleq f(\boldsymbol{x}_k) + \nabla f(\boldsymbol{x}_k)^T(\boldsymbol{x} - \boldsymbol{x}_k) + \frac{1}{2}(\boldsymbol{x} - \boldsymbol{x}_k)^T \nabla^2 f(\boldsymbol{x}_k)(\boldsymbol{x} - \boldsymbol{x}_k)$$

Minimizing quadratic approximation

$$\nabla \hat{f}(\boldsymbol{x}) = \nabla^2 f(\boldsymbol{x}_k)(\boldsymbol{x} - \boldsymbol{x}_k) + \nabla f(\boldsymbol{x}_k) = \boldsymbol{0}$$

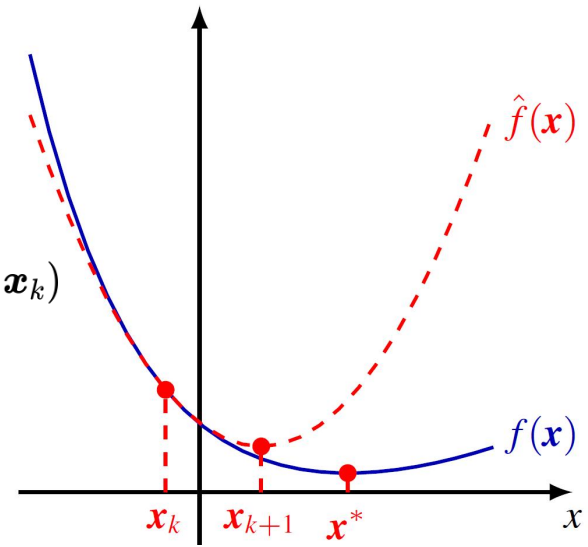
$$\Longrightarrow \boldsymbol{x} = \boldsymbol{x}_k - \left[\nabla^2 f(\boldsymbol{x}_k)\right]^{-1} \nabla f(\boldsymbol{x}_k)$$

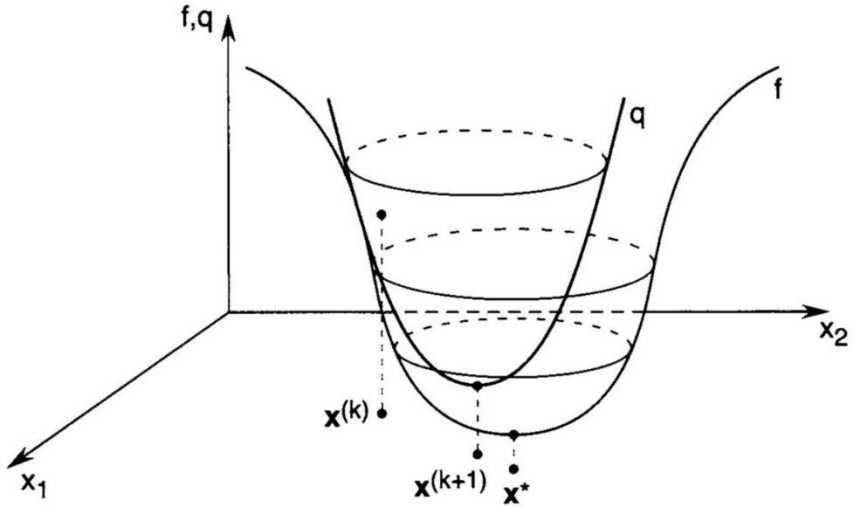
provided $\nabla^2 f(\boldsymbol{x}_k) \succ \boldsymbol{O}$

Newton step

$$\boldsymbol{x}_{k+1} = \boldsymbol{x}_k - \left[\nabla^2 f(\boldsymbol{x}_k)\right]^{-1} \nabla f(\boldsymbol{x}_k)$$

Note. If the function is quadratic, then Newton's method gets to the optimum in a single step.

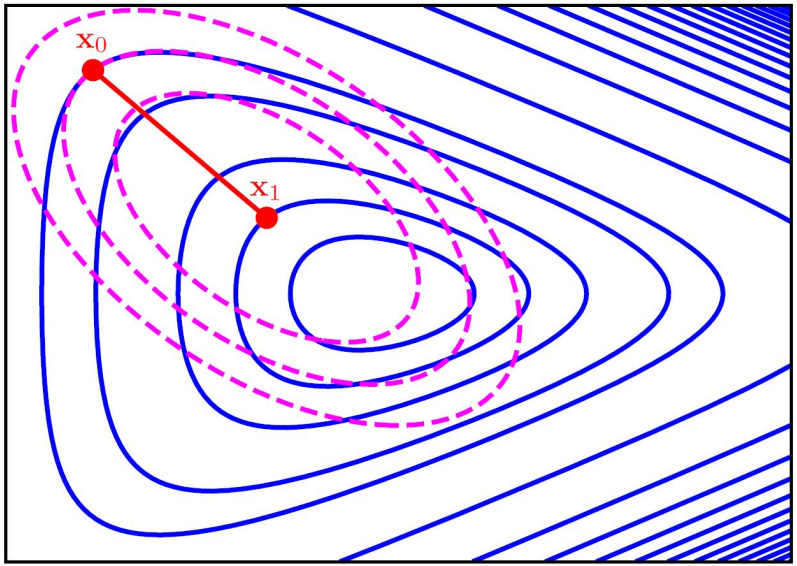


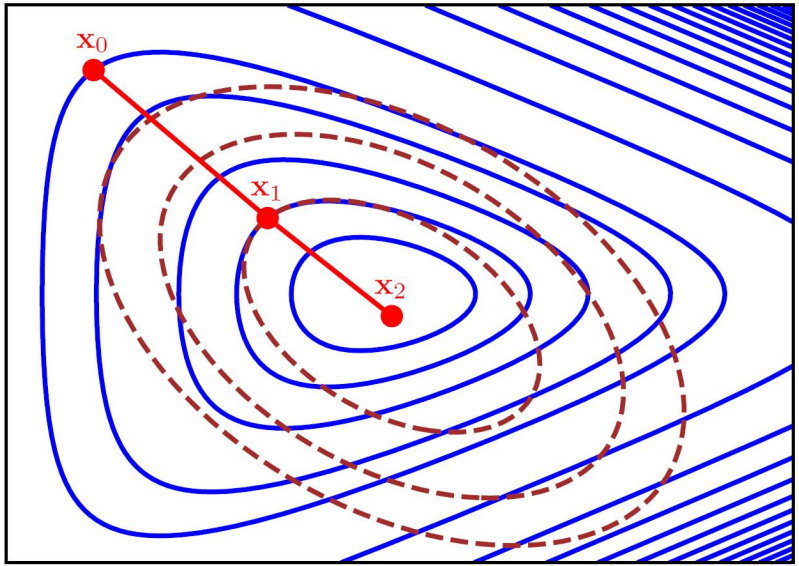
# Newton's Method

Example

The magenta curves are the level curves of the quadratic approximation at $\boldsymbol{x}_0$




The brown curves are the level curves of the quadratic approximation at $\boldsymbol{x}_1$



# Newton's Method

Example

$$f(x_1, x_2) = e^{x_1+3x_2-0.1} + e^{x_1-3x_2-0.1} + e^{-x_1-0.1}$$

Newton step at $\boldsymbol{x}_0 = (-2, 1)^T$

- gradient $$\nabla f(\boldsymbol{x}_0) = e^{-0.1} \begin{pmatrix} e^{x_1+3x_2} + e^{x_1-3x_2} - e^{-x_1} \\ 3e^{x_1+3x_2} - 3e^{x_1-3x_2} \end{pmatrix} \Bigg|_{\boldsymbol{x}=x_0} = \begin{pmatrix} -4.22019458 \\ 7.36051909 \end{pmatrix}$$

- Hessian $$\nabla^2 f(\boldsymbol{x}_0) = e^{-0.1} \begin{pmatrix} e^{x_1+3x_2} + e^{x_1-3x_2} + e^{-x_1} & 3e^{x_1+3x_2} - 3e^{x_1-3x_2} \\ 3e^{x_1+3x_2} - 3e^{x_1-3x_2} & 9e^{x_1+3x_2} + 9e^{x_1-3x_2} \end{pmatrix} \Bigg|_{x=x_0}$$
$$= \begin{pmatrix} 9.1515943 & 7.36051909 \\ 7.36051909 & 22.19129872 \end{pmatrix}$$

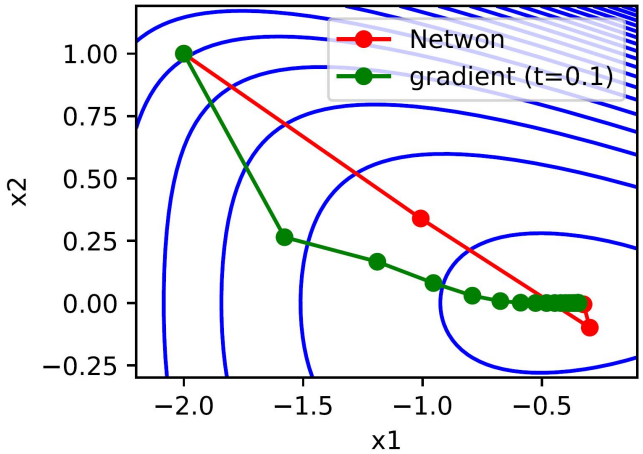
- Newton step
$$\boldsymbol{x}_1 = \boldsymbol{x}_0 - \left[\nabla^2 f(\boldsymbol{x}_0)\right]^{-1} \nabla f(\boldsymbol{x}_0) = \begin{pmatrix} -1.00725064 \\ 0.33903509 \end{pmatrix}$$

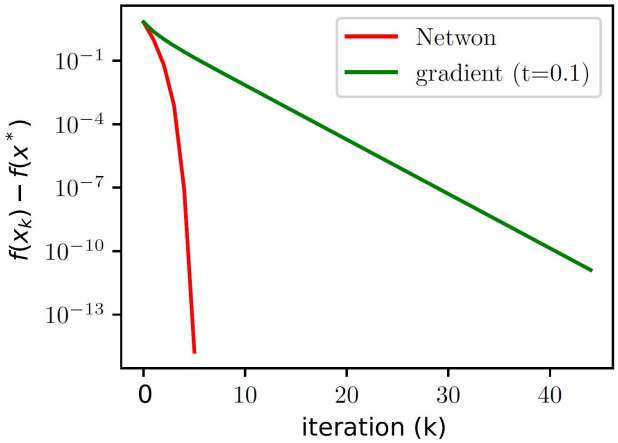
# Newton's Method

Example

$$f(x_1, x_2) = e^{x_1+3x_2-0.1} + e^{x_1-3x_2-0.1} + e^{-x_1-0.1}$$

Solution using Newton's method and gradient descent with constant step size 0.1




- Newton's method takes a more "direct" path
- Newton's method requires much fewer iterations, but each iteration is more expensive

# Newton's Method

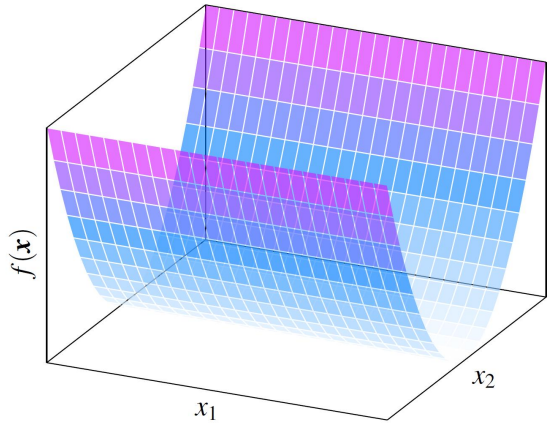
Three aspect to evaluate a numerical optimization method:

1. Convergence speed (How to measure the rate? In Lec2.).
2. Stability when applied to different functions.
3. Computation work per iteration.

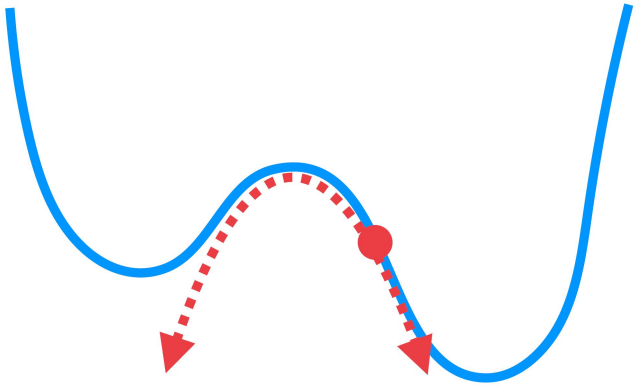
# Newton's Method

Drawbacks: In practice Hessian can be singular and indefinite

$$\boldsymbol{x}_{k+1} = \boldsymbol{x}_k - \left[\nabla^2 f(\boldsymbol{x}_k)\right]^{-1} \nabla f(\boldsymbol{x}_k)$$




Singular Hessian!



If start at a point of negative curvature, the Newton goes up!

(negative) search direction: must form an acute angle with the gradient

# Practical Newton's Method

initialization $\boldsymbol{x} \leftarrow \boldsymbol{x}_0 \in \mathbb{R}^n$

while $\|\nabla f(\boldsymbol{x})\| > \delta$ do

$\boldsymbol{d} \leftarrow -\boldsymbol{M}^{-1} \nabla f(\boldsymbol{x})$

$t \leftarrow$ backtracking line search

$x \leftarrow x + td$

end while

return

- Choose a positive-definite ***M*** that is close to the Hessian
- Solve the linear system via factorization instead of inversion
- line search does not need grad and Hessian

# Practical Newton's Method

$$\left[\nabla^2 f(\boldsymbol{x})\right]\boldsymbol{d} = -\nabla f(\boldsymbol{x})$$

is bad conditioned when Hessian is positive semi-definite (PSD) or indefinite.

- If function is convex, its Hessian must be PSD. We choose a ***M*** as

$$\boldsymbol{M} = \nabla^2 f(\boldsymbol{x}) + \epsilon \boldsymbol{I},\ \epsilon = \min(1, \|\nabla f(\boldsymbol{x})\|_\infty)/10$$

  Since ***M*** is PD, the search direction is solved by Cholesky factorization

$$\boldsymbol{M}\boldsymbol{d} = -\nabla f(\boldsymbol{x}),\ \boldsymbol{M} = \boldsymbol{L}\boldsymbol{L}^{\mathrm{T}}$$

  where ***L*** is a lower triangular matrix.

- If function is nonconvex, its Hessian can be indefinite. We compute ***M*** on the fly through

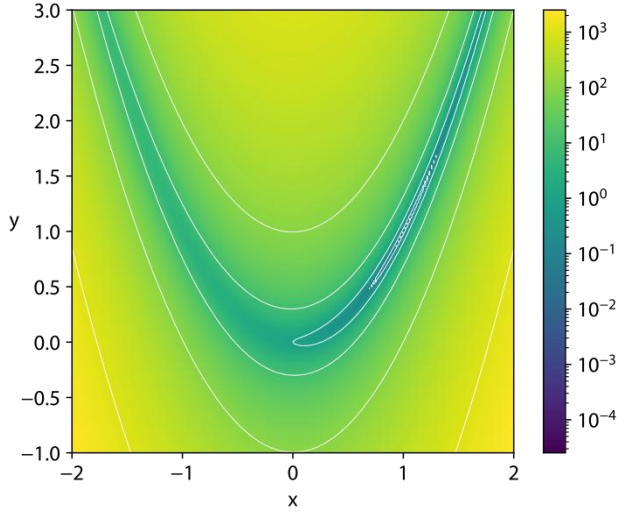
  Bunch-Kaufman Factorization

$$\boldsymbol{M}\boldsymbol{d} = -\nabla f(\boldsymbol{x}),\ \boldsymbol{M} = \boldsymbol{L}\boldsymbol{B}\boldsymbol{L}^{\mathrm{T}}$$

  where ***B*** is block diagonal matrix with block size 1x1 or 2x2. All 2x2 blocks contain nonpositive eigenvalues, which are easy to modify.

# Homework

Implement the line-search steepest gradient descent with Armijo condition.




$$f(\mathbf{x}) = f(x_1, x_2, \ldots, x_N) = \sum_{i=1}^{N/2} \left[ 100 \left( x_{2i-1}^2 - x_{2i} \right)^2 + \left( x_{2i-1} - 1 \right)^2 \right]$$

Your program should properly minimize this Rosenbrock function.

# Thanks for Listening!

*Ack.: The slide is prepared by Zhepei Wang and Fei Gao, improved by Lingfeng Wang, Song Zhao, and Shuo Yang.*