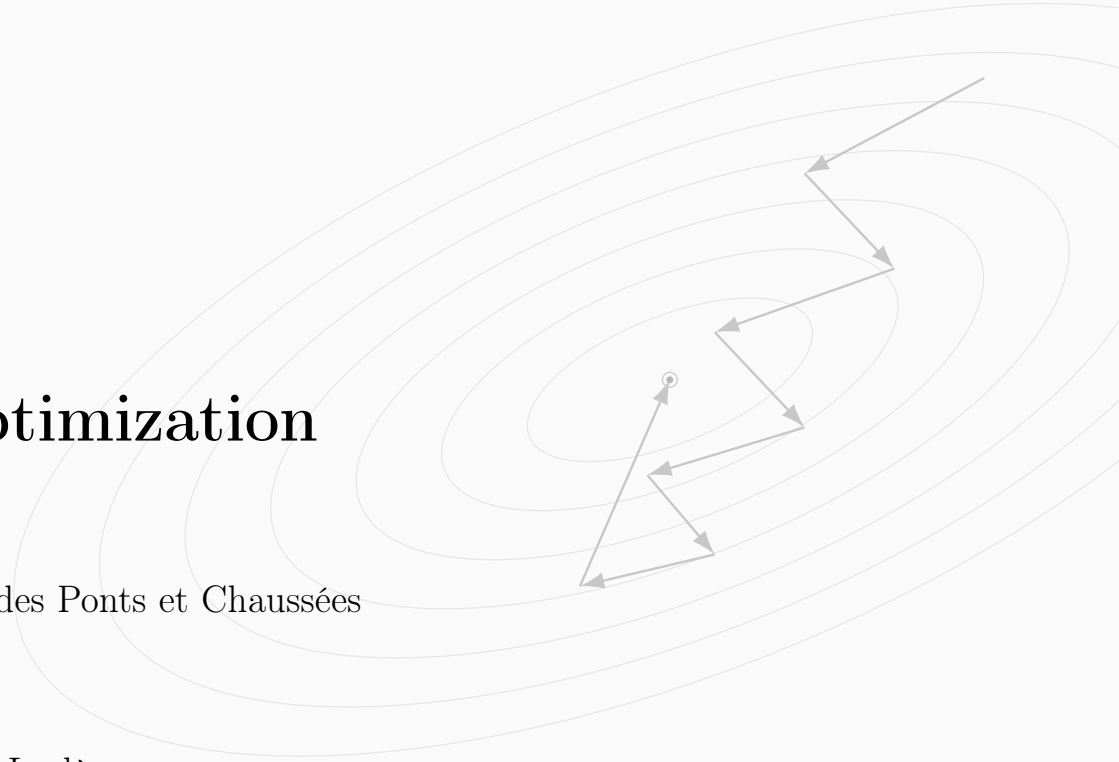


Convex Optimization

IMI – École Nationale des Ponts et Chaussées

Vincent Leclère

2026

CERMICS
École nationale des ponts et chaussées

ÉCOLE NATIONALE DES PONTS ET CHAUSSÉES

INSTITUT POLYTECHNIQUE DE PARIS

Contents

Linear Algebra for Optimization

V. Leclère (ENPC)

February 13th, 2026

What is this course about?

- **Goal:** build a complete toolkit to model, analyze, and solve optimization problems.
- **Foundations (why it works):**
  - numerical linear algebra for optimization;
  - convex geometry and convex analysis;
  - first-order optimality and constraint qualifications $\Rightarrow$ KKT and certificates.
- **Duality as a unifying lens:** Lagrangians, strong duality (Slater), multipliers as sensitivity prices, KKT $\Leftrightarrow$ saddle points (convex).
- **Algorithms (how to compute):**
  - gradient methods, steepest descent, conjugate gradient; rates and conditioning;
  - Newton and quasi-Newton (BFGS/L-BFGS): fast local accuracy with globalization;
  - constrained methods: a brief overview;
  - a quick look at stochastic gradient methods.
- **Throughout:** emphasize certificates (gaps, gradient mappings, dual bounds), assumptions, and when guarantees do or do not apply.

Course structure (12 sessions)

1. Numerical linear algebra
2. Convex sets
3. Convex functions
4. Optimality conditions
5. Convex duality
6. Optimization zoo
7. Gradient descent
8. Newton's method
9. Constrained optimization
10. Interior-point methods
11. Stochastic gradient
12. Exam

Why start with linear algebra?

- Your algorithms will be written in terms of norms, inner products, and linear systems.
- Convergence of Gradient Descent, Conjugate Gradient, Newton, Interior-Point Methods is essentially governed by spectra: $\lambda_{\min}, \lambda_{\max}$ and conditioning.
- In this course we will use mostly symmetric matrices (PSD order, spectral theorem), but we need Singular Value Decomposition (SVD) for operator norms and conditioning.
- Goal for today: fix a common vocabulary and a few key facts we will reuse all semester.

Convention in these slides

Icons / tags:

- ♡ : really important results
- ♢ : more advanced content
- ♣ : very simple in-class exercise
- ♠ : training exercise (harder)
- BV x.y : *Convex Optimization*, S. Boyd, L. Vanderberghe, Ch. x, Sec. y
- JCG x.y : *Fragments d'Optimisation Différentiable* (J.-Ch. Gilbert), Ch. x, Sec. y
- ➥ both are available online for free.
- ➥ Boyd's classes on convex optimization are on YouTube.

Color coding[a]:

- In text:
  - violet for definitions,
  - red for key results/theorems/assumptions.
- In math:
  - x primal variables,
  - λ multipliers,
  - θ parameters,
  - L smoothness constants,
  - $x^{(k)}$ iterates,
  - $\tau^{(k)}$ step sizes,
  - d directions.

[a] Best effort

Eigenvalues, eigenvectors, eigenspaces, spectrum (recall) ♡

Let $A \in \mathbb{R}^{n\times n}$ (we will always consider real matrices).

- A nonzero vector $x \neq 0$ is an eigenvector of A associated with eigenvalue $\lambda \in \mathbb{R}$ if

$$Ax = \lambda x.$$

- The eigenspace associated with λ is

$$E_\lambda(A) := \ker(A - \lambda I) = \{x \in \mathbb{R}^n : Ax = \lambda x\}.$$

- The spectrum of A is the set of its eigenvalues:

$$\mathrm{sp}(A) := \{\lambda \in \mathbb{R} : \det(A - \lambda I) = 0\}.$$

♣ Exercise: Show that A is invertible iff $0 \notin \mathrm{sp}(A)$.

Diagonalization (general case): definition and criterion

Let $A \in \mathbb{R}^{n\times n}$.

- A is diagonalizable (over $\mathbb{R}$) if there exist an invertible matrix P and a diagonal matrix D such that

$$A = PDP^{-1}.$$

- In that case, the diagonal entries of D are eigenvalues of A, and the columns of P form a basis of eigenvectors.

Diagonalization criterion (via eigenspaces)

A is diagonalizable over $\mathbb{R}$ iff:

1. all eigenvalues are real (so $\mathrm{sp}(A) \subset \mathbb{R}$), and
2. the eigenvectors span $\mathbb{R}^n$, equivalently

$$n = \sum_{\lambda \in \mathrm{sp}(A)} \dim\left(E_\lambda(A)\right).$$

♣ Exercise: Assume A has n *distinct* real eigenvalues. Show that A is diagonalizable.

(Semi)definite matrices and PSD order

Let S_n be the set of real symmetric $n \times n$ matrices.

- Positive semidefinite (PSD): $A \succeq 0$ (equivalently $A \in S_n^+$) iff

$$x^\top Ax \geq 0 \quad \forall x \in \mathbb{R}^n.$$

- Positive definite (PD): $A \succ 0$ (equivalently $A \in S_n^{++}$) iff

$$x^\top Ax > 0 \quad \forall x \neq 0.$$

Loewner (PSD) order on S_n

For $A, B \in S_n$, define

$$A \preceq B \iff B - A \succeq 0.$$

Equivalently, $A \preceq B$ iff $x^\top Ax \leq x^\top Bx$ for all $x \in \mathbb{R}^n$.

♣ Exercise: Show that $\preceq$ is a partial order on S_n (reflexive, antisymmetric, transitive).

Spectral theorem (real symmetric matrices) ♡

Let S_n be the set of real symmetric $n \times n$ matrices.

Theorem (Spectral theorem)

Any symmetric matrix $A \in S_n$ has n real eigenvalues (counted with multiplicity) $\lambda_1, \dots, \lambda_n$ and an orthonormal basis of eigenvectors $(q_i)_{i \in [n]}$ i.e.,such that:

$$Aq_i = \lambda_i q_i, \quad \text{and} \quad q_i^\top q_j = \delta_{ij}, \qquad \forall i, j.$$

In other words, there exists an orthogonal matrix Q (i.e. $Q^\top Q = I$) such that

$$A = Q \Lambda Q^\top,$$

where $\Lambda = \mathrm{diag}(\lambda_1, \dots, \lambda_n)$ and $\lambda_1, \dots, \lambda_n$ are the eigenvalues of A.

Consequences for (semi)definite matrices ♡

For $A \in S_n$ with spectral decomposition $A = Q\Lambda Q^\top$, where $\Lambda = \mathrm{diag}(\lambda_1, \dots, \lambda_n)$:

- $A \in S_n^+ \iff A \succeq 0 \iff \lambda_i \geq 0$ for all i.
- $A \in S_n^{++} \iff A \succ 0 \iff \lambda_i > 0$ for all i.
- For any $A \in S_n^+$, with $A = Q\Lambda Q^\top$, we have

$$A^{1/2} = Q\Lambda^{1/2}Q^\top, \quad \text{where} \quad \Lambda^{1/2} = \mathrm{diag}(\sqrt{\lambda_1}, \dots, \sqrt{\lambda_n}),$$

  where $A^{1/2}$ is the unique PSD square root of A (i.e. $A^{1/2}A^{1/2} = A$).

♠ Exercise: Prove the above statements using the spectral decomposition.

Rayleigh quotient and eigenvalue extrema ♢

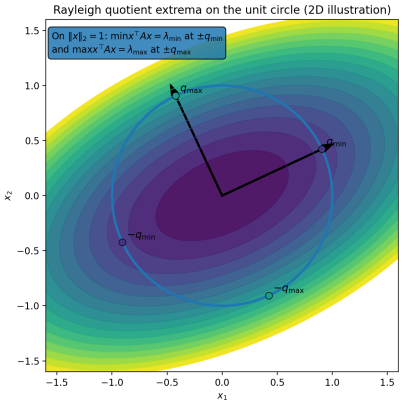
For $A \in S_n$ and $x \neq 0$, define $R_A(x) = \frac{x^\top Ax}{x^\top x}$.

- $\lambda_{\min}(A) = \min_{\|x\|_2 = 1} x^\top Ax$,
  $\lambda_{\max}(A) = \max_{\|x\|_2 = 1} x^\top Ax$.

➥ min and max are attained at eigenvectors associated with $\lambda_{\min}(A)$ and $\lambda_{\max}(A)$.

- If $A \succ 0$, then $\|x\|_A^2 = x^\top Ax$ satisfies

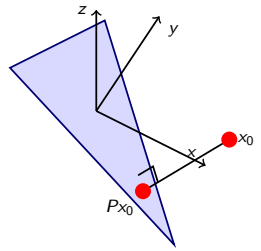
$$\lambda_{\min}(A)\,\|x\|_2^2 \leq \|x\|_A^2 \leq \lambda_{\max}(A)\,\|x\|_2^2.$$



Orthogonal projectors

A matrix P is an orthogonal projector iff $P^2 = P$ and $P = P^\top$.

- Then $\mathbb{R}^n = \mathrm{Im}(P) \oplus \ker(P)$, and Px is the closest (for $\|\cdot\|_2$) point to x in $\mathrm{Im}(P)$.
- Eigenvalues of P are 0 or 1, so $P \succeq 0$ and $\|P\|_2 = 1$ (unless $P = 0$).
- If $\|q\|_2 = 1$, then $P = qq^\top$ is the orthogonal projector onto $\mathrm{span}(q)$.
- More generally, if $Q \in \mathbb{R}^{n \times k}$ has orthonormal columns, then $P = QQ^\top$ projects onto $\mathrm{Im}(Q)$.



Px_0 is the Euclidean closest point to x_0 in $\mathrm{Im}(P)$.

Spectral decomposition as a sum of rank-one matrices

Let $A \in S_n$ with spectral decomposition $A = Q\Lambda Q^\top$, where $Q = [q_1 \cdots q_n]$ is orthogonal and $\Lambda = \mathrm{diag}(\lambda_1, \ldots, \lambda_n)$.

Rank-one expansion

$$A = \sum_{i=1}^{n} \lambda_i \, q_i q_i^\top.$$

Each term $q_i q_i^\top$ is the orthogonal projector onto $\mathrm{span}(q_i)$ (rank one).

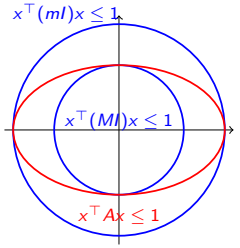
Explanation: Since $Q\Lambda Q^\top = \sum_{i=1}^n \lambda_i \, Q e_i e_i^\top Q^\top$ and $Qe_i = q_i$,

$$Q\Lambda Q^\top = \sum_{i=1}^{n} \lambda_i \, q_i q_i^\top.$$

PSD order and basic algebra

For $A, B \in S_n$, recall that $A \preceq B$ iff $B - A \succeq 0$.

- Order is compatible with quadratic forms: $A \preceq B \Rightarrow x^\top A x \leq x^\top B x$ for all x.
- If $0 \prec A \preceq B$, then $B^{-1} \preceq A^{-1}$ (order reverses under inversion).
- $A \preceq B$ implies $\lambda_{\min}(A) \leq \lambda_{\min}(B)$ and $\lambda_{\max}(A) \leq \lambda_{\max}(B)$.
- ➥ $A \preceq MI$ iff $\lambda_{\max}(A) \leq M$, and $A \succeq mI$ iff $\lambda_{\min}(A) \geq m$.
- ➥ We will constantly use $mI \preceq \nabla^2 f(x) \preceq MI$ to control GD and Newton.



Norms: a quick recall ♡

A norm on $\mathbb{R}^n$ is a map $\|\cdot\| : \mathbb{R}^n \to \mathbb{R}_+$ such that:

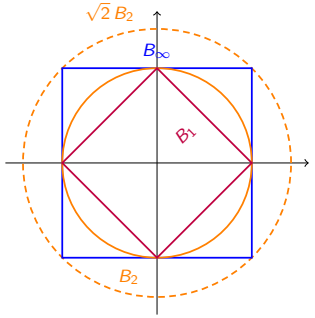
- Positive definiteness: $\|x\| \geq 0$ and $\|x\| = 0 \iff x = 0$.
- Absolute homogeneity: $\|\alpha x\| = |\alpha| \, \|x\|$ for all $\alpha \in \mathbb{R}$.
- Triangle inequality: $\|x + y\| \leq \|x\| + \|y\|$.
- Classical examples:

$$\|x\|_2 = \sqrt{\sum_i x_i^2}, \quad \|x\|_1 = \sum_i |x_i|, \quad \|x\|_\infty = \max_i |x_i|.$$

- We say that two norms $\|\cdot\|_a$ and $\|\cdot\|_b$ are equivalent if there exist constants $c, C > 0$ such that

$$c \, \|\cdot\|_a \leq \|\cdot\|_b \leq C \, \|\cdot\|_a.$$

- In finite dimension, all norms are equivalent!



Useful inequalities (for $x \in \mathbb{R}^n$):

$$\|x\|_\infty \leq \|x\|_2 \leq \|x\|_1 \leq \sqrt{n} \, \|x\|_2.$$

Inner products and geometry

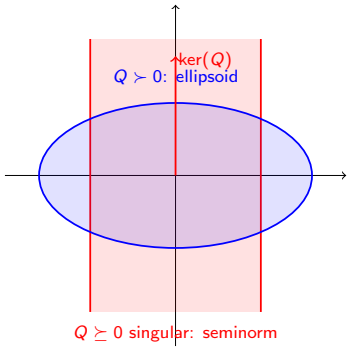
- An inner product on $\mathbb{R}^n$ is a map $\langle \cdot, \cdot \rangle : \mathbb{R}^n \times \mathbb{R}^n \to \mathbb{R}$ such that:
  - Bilinear: $\langle ax_1 + bx_2, \, y \rangle = a\langle x_1, y \rangle + b\langle x_2, y \rangle$ and $\langle x, \, ay_1 + by_2 \rangle = a\langle x, y_1 \rangle + b\langle x, y_2 \rangle$.
  - Symmetric: $\langle x, y \rangle = \langle y, x \rangle$.
  - Positive definite: $\langle x, x \rangle \geq 0$ and $\langle x, x \rangle = 0 \iff x = 0$.
- It defines a norm: $\|x\| := \sqrt{\langle x, x \rangle}$.
- Cauchy–Schwarz: $|\langle x, y \rangle| \leq \|x\| \, \|y\|$.
  - Equality case: if $x \neq 0$ and $y \neq 0$, equality holds iff $y = \alpha x$ for some $\alpha \in \mathbb{R}$ (colinearity).
  - (Trivial cases: if $x = 0$ or $y = 0$, equality holds with both sides $= 0$.)
- Orthogonality depends on the inner product: $x \perp y \iff \langle x, y \rangle = 0$.

The $\|\cdot\|_Q$ norm (and when it is not a norm)

Let $Q \in S_n^+$.

- Define $\|x\|_Q := \sqrt{x^\top Q x}$.
- If $Q \succ 0$: $\|\cdot\|_Q$ is a norm and its unit ball is an ellipsoid.
- If $Q \succeq 0$ but singular: $\|\cdot\|_Q$ is a seminorm, with nullspace $\ker(Q) = \{x : Qx = 0\}$.
- Change of variables (SPD case): $\|x\|_Q = \|Q^{1/2}x\|_2$.

♠ Exercise: Show $\|x\|_Q \le \sqrt{\lambda_{\max}(Q)}\,\|x\|_2$ and $\|x\|_2 \le \frac{1}{\sqrt{\lambda_{\min}(Q)}}\,\|x\|_Q$ when $Q \succ 0$.



Unit "ball": ellipse (SPD) vs unbounded strip (singular PSD).

Gradient (recap)

Let $f : \mathbb{R}^n \to \mathbb{R}$ be differentiable at x.

Gradient = the unique vector defining the first-order approximation

There exists a unique vector $g \in \mathbb{R}^n$ such that

$$f(x+h) = f(x) + g^\top h + o(\|h\|_2) \qquad \text{as } h \to 0.$$

We denote it $g = \nabla f(x)$.

Coordinate formula

In the canonical basis,

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

Hessian and second-order Taylor expansion (recap)

If f is twice differentiable at x:

Hessian = the unique matrix defining the second-order approximation

There exists a unique symmetric matrix $H \in S_n$ such that

$$f(x+h) = f(x) + \nabla f(x)^\top h + \tfrac{1}{2}\, h^\top H h + o(\|h\|_2^2) \qquad \text{as } h \to 0.$$

We denote it $H = \nabla^2 f(x)$.

Coordinate formula

$$\nabla^2 f(x) = \left[\frac{\partial^2 f}{\partial x_i\, \partial x_j}(x)\right]_{i,j=1}^{n}.$$

Hessian bounds and strong convexity

Assume $f : \mathbb{R}^n \to \mathbb{R}$ is twice differentiable.

PSD order $\iff$ eigenvalue bounds

For any symmetric matrix $H \in S_n$ and any $m \in \mathbb{R}$,

$$H \succeq mI \iff \lambda_{\min}(H) \ge m \iff x^\top H x \ge m\|x\|_2^2 \;\; \forall x \in \mathbb{R}^n.$$

Strong convexity via the Hessian

If there exists $m > 0$ such that

$$\nabla^2 f(x) \succeq mI \qquad \forall x \in \mathbb{R}^n,$$

then f is m-strongly convex (w.r.t. $\|\cdot\|_2$), i.e.

$$f(y) \ge f(x) + \nabla f(x)^\top (y-x) + \frac{m}{2}\|y-x\|_2^2 \qquad \forall x, y \in \mathbb{R}^n.$$

Equivalently, $\lambda_{\min}(\nabla^2 f(x)) \ge m$ for all x.

Dual norm: definition and two key examples

Given a norm $\|\cdot\|$, its dual norm is

$$\|y\|_\star := \sup_{\|x\|\le 1} y^\top x.$$

- Generalized Cauchy-Schwarz: $y^\top x \le \|y\|_\star \|x\|$.
- Examples:
  - $(\|\cdot\|_2)^\star = \|\cdot\|_2$,
  - $(\|\cdot\|_1)^\star = \|\cdot\|_\infty$,
  - $(\|\cdot\|_\infty)^\star = \|\cdot\|_1$.

Induced operator norm and the spectral norm

For matrix A, the operator norm induced by $\|\cdot\|_{op}$ is

$$\|A\|_{op} := \sup_{x\neq 0} \frac{\|Ax\|}{\|x\|}.$$

- For $\|\cdot\|_2$: $\|A\|_{2,op}$ is the spectral norm.
- Equivalent characterizations:
  - $\|A\|_{2,op} = \sqrt{\lambda_{\max}(A^\top A)}$.
  - $\|A\|_{2,op} = \sigma_{\max}(A)$ (largest singular value – see later).
- For $A \in S_n$: $\|A\|_{2,op} = \max_i |\lambda_i(A)|$
- For $A \in S_n^+$, $\|A\|_{2,op} = \lambda_{\max}(A)$.

Frobenius norm (matrix "Euclidean" norm)

For $A \in \mathbb{R}^{m\times n}$, define

$$\|A\|_F := \sqrt{\sum_{i,j} A_{ij}^2} = \sqrt{\mathrm{Tr}(A^\top A)}.$$

If $A = U\Sigma V^\top$ is an SVD with singular values (σ_i), then

$$\|A\|_F^2 = \sum_{i=1}^{p} \sigma_i^2, \qquad \|A\|_2 \le \|A\|_F \le \sqrt{\mathrm{rank}(A)}\,\|A\|_2.$$

Preview: Newton as repeated quadratic minimization

Let $f : \mathbb{R}^n \to \mathbb{R}$ be twice differentiable. Use second order Taylor expansion around the current iterate $x^{(k)}$:

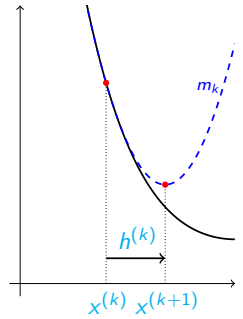
$$f(x^{(k)} + h) = m_k(h) + o(\|h\|_2^2),$$
$$m_k(h) := f(x^{(k)}) + \nabla f(x^{(k)})^\top h + \tfrac{1}{2} h^\top \nabla^2 f(x^{(k)})\, h.$$

Newton idea

Newton[a] is: build a quadratic model, minimize it, repeat.

$$h^{(k)} \leftarrow \arg\min_h m_k(h)$$
$$x^{(k+1)} = x^{(k)} + h^{(k)}$$

[a] Often the fastest *local* method under strong regularity assumptions.



Newton step as quadratic model minimization

Minimizing the quadratic model gives a linear system

Recall the local quadratic model at $x^{(k)}$:

$$m_k(h) := f(x^{(k)}) + \nabla f(x^{(k)})^\top h + \tfrac{1}{2}\, h^\top \nabla^2 f(x^{(k)})\, h.$$

Optimality condition

A minimizer $h^{(k)}$ satisfies $\nabla m_k(h^{(k)}) = 0$. Since

$$\nabla m_k(h) = \nabla f(x^{(k)}) + \nabla^2 f(x^{(k)})\, h,$$

we obtain the (symmetric) linear system

$$\nabla^2 f(x^{(k)})\, h^{(k)} = -\nabla f(x^{(k)}).$$

If $\nabla^2 f(x^{(k)}) \succ 0$, the minimizer is unique.

How to solve $Ax = b$?

Goal: solve one linear system $Ax = b$.

Triangular systems are easy

If L is lower triangular, $Lx = b$ is solved by forward substitution (one pass).
If U is upper triangular, $Ux = b$ is solved by backward substitution (one pass).
$\rightsquigarrow O(n^2)$ operations each.

Computing A^{-1} means solving n systems

Let $e_1, \dots, e_n$ be the canonical basis. Then

$$A^{-1} = \begin{bmatrix} x^{(1)} & \cdots & x^{(n)} \end{bmatrix} \quad \text{with} \quad Ax^{(i)} = e_i \ \ (i = 1, \dots, n).$$

So forming A^{-1} amounts to solving n right-hand sides.
$\rightsquigarrow O(n^3)$ operations total.

Conclusion: solve $Ax = b$ with dedicated tools, do not invert.

Why factorize?

In optimization, we repeatedly solve linear systems $Ax = b$ (Newton steps, least-squares, KKT blocks, ...).

Principle

Factorize once, solve many times.

- We reuse the same factorization, we do not compute A^{-1}.
- The expensive part is the factorization; each new right-hand side is cheap (triangular solves).

♣ Exercise: Give two different vectors $b^{(1)}, b^{(2)}$. Explain why reusing the same factorization of A is efficient.

LU factorization (square matrices)

For a general square matrix $A \in \mathbb{R}^{n \times n}$, we aim for

$$PA = LU,$$

where L is lower triangular (often with ones on the diagonal) and U is upper triangular, and P is a permutation matrix (pivoting for numerical stability).

♣ Exercise: Assume $A = LU$. Show that solving $Ax = b$ reduces to two triangular solves. Show that it stays the same if $PA = LU$.

Cholesky factorization (symmetric positive definite) ♢

If $A \in S_n^{++}$ (*i.e.*,$A \succ 0$), then there exists a unique lower triangular L with positive diagonal such that

$$A = LL^\top.$$

- Solve $Ax = b$ via:

$$Ly = b, \qquad L^\top x = y.$$

- This is the standard path for Hessian systems in strongly convex optimization.

♣ Exercise: Show that if $A = LL^\top$ and L is invertible, then $A \succ 0$.

Why $LL^\top$ (Cholesky) over LU? ♢

Assume $A \in S_n$ and $A \succ 0$.

- Faster (dense): Cholesky costs about $\frac{1}{3}n^3$ flops vs about $\frac{2}{3}n^3$ for LU.
- Less memory: store essentially one triangular factor (symmetry is exploited).
- More robust here: for SPD matrices, Cholesky does not require pivoting.
- Matches optimization geometry: $x^\top Ax = \|L^\top x\|_2^2$.

Worked example: Cholesky on a 2×2 SPD matrix ♢

Let

$$A = \begin{pmatrix} 4 & 2 \\ 2 & 3 \end{pmatrix}, \qquad b = \begin{pmatrix} 2 \\ 1 \end{pmatrix}.$$

We look for $A = LL^\top$ with $L = \begin{pmatrix} \ell_{11} & 0 \\ \ell_{21} & \ell_{22} \end{pmatrix}$.

Matching entries:

$$\ell_{11} = 2, \qquad \ell_{21} = 1, \qquad \ell_{22} = \sqrt{2}, \quad \Rightarrow \quad L = \begin{pmatrix} 2 & 0 \\ 1 & \sqrt{2} \end{pmatrix}.$$

Solve $Ly = b$: $y_1 = 1$, $y_2 = 0$. Then solve $L^\top x = y$: $x_2 = 0$, $x_1 = \frac{1}{2}$.

$$x = \begin{pmatrix} \frac{1}{2} \\ 0 \end{pmatrix}.$$

♣ Exercise: Change b to $\begin{pmatrix} 0 \\ 1 \end{pmatrix}$. Reuse the same L and solve again.

Takeaways (what to remember)

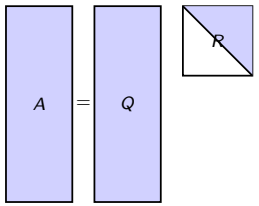
- Solving $Ax = b$ is done via factorization + triangular solves, not via A^{-1}.
- Use Cholesky when A is symmetric positive definite: $A = LL^\top$.
- Use LU as the general-purpose tool (often with a permutation $PA = LU$).
- Reuse factorizations whenever the matrix A stays the same and only b changes.

♠ Exercise: (Optimization link) Consider $f(x) = \frac{1}{2}x^\top Ax - b^\top x$ with $A \succ 0$. Show that the minimizer solves $Ax = b$, and explain how Cholesky gives the minimizer efficiently.

QR factorization (thin QR)

For a matrix $A \in \mathbb{R}^{m\times n}$, with $m \geq n$, there exist matrices $Q \in \mathbb{R}^{m\times n}$ and $R \in \mathbb{R}^{n\times n}$ such that

$$A = QR,$$
$$Q^\top Q = I_n,$$
$$R \text{ upper triangular.}$$



tall matrix $m \geq n$

Blue = potentially nonzero entries .

In particular:

- Q has orthonormal columns. If $\mathrm{rank}(A) = n$, they form a basis of $\mathrm{Im}(A)$.
- R is a change of coordinates in that basis (triangular $\Rightarrow$ easy solves).
- If A has full column rank, then R is invertible.
- Note that $P := QQ^\top$ is the orthogonal projector onto $\mathrm{Im}(Q)$.

Least squares and why we avoid normal equations

Given $A \in \mathbb{R}^{m\times n}$ and $b \in \mathbb{R}^m$, the (linear) least-squares problem is

$$\min_{x\in\mathbb{R}^n} \|Ax - b\|_2^2.$$

Optimality condition (normal equations)

If A has full column rank, the objective is differentiable and strictly convex, and the unique minimizer $x^\star$ satisfies

$$A^\top(Ax^\star - b) = 0 \quad \Longleftrightarrow \quad A^\top A x^\star = A^\top b.$$

But: normal equations can be numerically bad

Forming $A^\top A$ typically squares the conditioning:

$$\kappa_2(A^\top A) = \kappa_2(A)^2 \quad \text{(when } A \text{ has full column rank).}$$

Conclusion: solve LS via QR (or SVD), not via $A^\top A$.

Solving least squares via QR (thin QR)

Assume $m \geq n$. Let $A = QR$ be a thin QR factorization with

$$Q \in \mathbb{R}^{m\times n}, \quad Q^\top Q = I_n, \qquad R \in \mathbb{R}^{n\times n} \text{ upper triangular.}$$

Key step: orthogonal decomposition

Let $P := QQ^\top$ (orthogonal projector onto $\mathrm{Im}(Q)$). Then for any x,

$$\|Ax - b\|_2^2 = \|QRx - b\|_2^2 = \|Rx - Q^\top b\|_2^2 + \|(I - P)b\|_2^2.$$

Consequence (full column rank)

If A has full column rank, then R is invertible and the unique minimizer satisfies

$$Rx^\star = Q^\top b,$$

solved by backward substitution.

Singular Value Decomposition (SVD): definition

SVD extends spectral decomposition to rectangular matrices.

Existence (main requirement)

For any $A \in \mathbb{R}^{m\times n}$, there exist orthogonal matrices $U \in \mathbb{R}^{m\times m}$, $V \in \mathbb{R}^{n\times n}$, and a rectangular diagonal matrix $\Sigma \in \mathbb{R}^{m\times n}$ such that

$$A = U\Sigma V^\top, \qquad \Sigma = \begin{cases} \mathrm{diag}(\sigma_1, \dots, \sigma_n) & \text{if } m \geq n, \\ \mathrm{diag}(\sigma_1, \dots, \sigma_m) & \text{if } m < n, \end{cases}$$

with $p = \min(m, n)$ and $\sigma_1 \geq \cdots \geq \sigma_p \geq 0$.

- U, V: orthogonal changes of basis (rotations/reflections).
- Σ: axis scalings; zeros encode directions collapsed by A.
- If $r = \mathrm{rank}(A)$: $\sigma_1 \geq \cdots \geq \sigma_r > 0$ and $\sigma_{r+1} = \cdots = \sigma_p = 0$.
- Geometry: $A(B_2)$ is an ellipsoid with semi-axes σ_i (directions = columns of U).

SVD: key properties

Let $A = U\Sigma V^\top$ be an SVD with singular values $\sigma_1 \geq \cdots \geq \sigma_p$.

- σ_i^2 are the positive eigenvalues of $A^\top A$ (and of $AA^\top$).
- $\|A\|_2 = \sigma_1$ (spectral/operator norm).
- $$\min_{\|x\|_2=1} \|Ax\|_2 = \begin{cases} \sigma_p & \text{if } \ker(A) = \{0\} \quad \text{(full column rank)}, \\ 0 & \text{otherwise.} \end{cases}$$
- If A is square invertible: $\|A^{-1}\|_2 = 1/\sigma_n$ and $\kappa_2(A) = \sigma_1/\sigma_n$.
- The truncated SVD gives the best rank-k approximation of A in spectral and Frobenius norms.

Condition number and why optimization cares

- For A invertible, $\kappa_2(A) := \|A\|_2 \|A^{-1}\|_2 = \sigma_1/\sigma_n \geq 1$.
- For $A \in S_n^{++}$: $\kappa_2(A) = \lambda_{\max}(A)/\lambda_{\min}(A)$.
- Linear systems: relative error can be amplified by $\kappa_2(A)$.
- GD on quadratics: rates depend on κ.

Preview: CG improves κ to $\sqrt{\kappa}$ in the convergence factor.

Stability vs conditioning (linear solves)

Two different notions:

- Conditioning = sensitivity of the *problem*.

  $Ax = b \quad \Rightarrow \quad$ how much does x change if b changes a bit?
- Stability = behavior of the *algorithm* in finite precision.

  does the computed $\hat{x}$ solve a nearby linear system?

Rule of thumb

Even with a very good algorithm, the best relative accuracy you can expect is often on the order of

$$\frac{\|\hat{x} - x\|}{\|x\|} \approx O(\kappa_2(A)\, u)\,,$$

where u is machine precision.

♣ Exercise: Take $A = \mathrm{diag}(1, \varepsilon)$, $b = (1, 1)$. What happens to the solution when ε is very small? Interpret with $\kappa_2(A)$.

Backward stability (the ideal guarantee)

Definition (informal)

An algorithm for solving $Ax = b$ is backward stable if it returns $\hat{x}$ such that

$$(A + \Delta A)\,\hat{x} = b \quad \text{with} \quad \frac{\|\Delta A\|}{\|A\|} \text{ small (typically } O(u)).$$

- Meaning: the algorithm behaves as if it solved *exactly* a slightly perturbed problem.
- If the problem is well-conditioned, a small backward error implies a small forward error.

Practical takeaway

Good solvers are designed to be (close to) backward stable:

- SPD $\Rightarrow$ Cholesky + triangular solves.
- General $A \Rightarrow$ LU with pivoting.
- Least-squares $\Rightarrow$ QR (avoid normal equations).

What you have to know

- Definitions: eigenvalues/eigenvectors; spectrum; PSD/PD; Loewner order $A \preceq B$.
- How to read $mI \preceq A \preceq MI$ as a "geometry sandwich" (ellipsoid between balls) and as eigenvalue bounds.
- Spectral theorem for symmetric matrices: $A = Q\Lambda Q^\top$ and consequences ($A \succeq 0 \Leftrightarrow \lambda_i \geq 0$).
- Dual norms and induced operator norm; in particular $\|A\|_2 = \sigma_{\max}(A)$ and $\|A\|_2 = \max_i |\lambda_i(A)|$ for $A \in S_n$.

What you really should know

- Rayleigh quotient facts: $\lambda_{\min} = \min_{\|x\|_2=1} x^\top Ax$, $\lambda_{\max} = \max_{\|x\|_2=1} x^\top Ax$.
- The projector viewpoint: $P^2 = P$, $P = P^\top$, Px is the Euclidean closest point in $\mathrm{Im}(P)$; $P = QQ^\top$.
- Conditioning as the main complexity parameter: $\kappa_2(A) = \lambda_{\max}/\lambda_{\min}$ (SPD), impacts GD/CG/Newton and numerical accuracy.
- Stability vs conditioning: good solvers aim for (near) backward stability; forward error typically scales like $\kappa \cdot u$.

What you have to be able to do

- Use spectral decomposition to prove PSD/PD statements and derive inequalities like $\lambda_{\min}\|x\|_2^2 \leq x^\top Ax \leq \lambda_{\max}\|x\|_2^2$.
- Recognize when $\|\cdot\|_Q$ is a norm vs a seminorm; use $\|x\|_Q = \|Q^{1/2}x\|_2$ when $Q \succ 0$.
- Solve $Ax = b$ efficiently via factorizations (LU / Cholesky): factorize once, then triangular solves for new right-hand sides.

What you should be able to do

- Diagnose ill-posedness: interpret tiny singular values / large κ as instability risk; know when SVD is the right diagnostic tool.
- Give quick "method choices" for linear algebra primitives: SPD $\rightarrow$ Cholesky; general square $\rightarrow$ pivoted LU; LS $\rightarrow$ QR (or SVD).

Convexity

V. Leclère (ENPC)

February 20th and 27th, 2026

Why should I bother to learn this stuff?

- Convex vocabulary and results are needed throughout the course, especially to obtain optimality conditions and duality relations.
- Convex analysis tools like Fenchel transform appears in modern machine learning theory
- $\Longrightarrow$ fundamental for M2 in continuous optimization
- $\Longrightarrow$ useful for M2 in operation research, machine learning (and some part of probability or mechanics)

Affine sets ♡

Let X be a normed vector space (usually $X = \mathbb{R}^n$), and $C \subset X$

- C is affine if it contains the entire line through any two distinct points of C, *i.e.,*

$$\forall x, y \in C, \quad \forall \theta \in \mathbb{R}, \qquad \theta x + (1-\theta) y \in C.$$

- The affine hull of C is the set of affine combination of elements of C,

$$\mathrm{aff}(C) := \left\{ \sum_{i=1}^{K} \theta_i x_i \;\middle|\; \forall x_i \in C,\ \forall \theta_i \in \mathbb{R},\ \sum_{i=1}^{K} \theta_i = 1,\ \forall i \in [K], \forall K \in \mathbb{N} \right\}$$

- $\mathrm{aff}(C)$ is the smallest affine space containing C.
- The affine dimension of C is the dimension of $\mathrm{aff}(C)$ (*i.e.,*the dimension of the vector space $\mathrm{aff}(C) - x_0$ for $x_0 \in C$).
- The relative interior of C is defined as

$$\mathrm{ri}(C) := \left\{ x \in C \;\middle|\; \exists r > 0, \quad B(x, r) \cap \mathrm{aff}(C) \subset C \right\}$$

Convex sets ♡

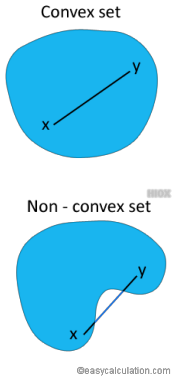
- C is convex if for any two points x and y in C the segment $[x, y] \subset C$, *i.e.,*

$$\forall x, y \in C,\ \forall \theta \in [0,1],\ \theta x + (1-\theta) y \in C.$$

- The convex hull of C is the set of convex combination of elements of C, *i.e.,*

$$\mathrm{conv}(C) := \left\{ \sum_{i=1}^{K} \theta_i x_i \mid \forall x_i \in C, \ \forall \theta_i \in [0,1],\ \sum_{i=1}^{K} \theta_i = 1,\ \forall i \in [K],\ \forall K \in \mathbb{N} \right\}$$

- $\mathrm{conv}(C)$ is the smallest convex set containing C.



Cones ♡

- C is a cone if for all $x \in C$ the ray $\mathbb{R}_+ x \subset C$, *i.e.*,

$$\forall x \in C, \quad \forall \theta \in \mathbb{R}_+, \qquad \theta x \in C.$$

- The (convex) conic hull of C is the set of all conic combination of elements of C *i.e.*,

$$\mathrm{cone}(C) := \left\{ \sum_{i=1}^{K} \theta_i x_i \mid \ \forall x_i \in C,\ \forall \theta_i \in \mathbb{R}_+,\ \forall i \in [K],\ \forall K \in \mathbb{N} \right\}$$

- $\mathrm{cone}(C)$ is the smallest convex cone containing C.
- A cone C is pointed if it does not contain any full line $\mathbb{R}x$ for $x \neq 0$.
- For C convex, $\mathrm{cone}(C) = \bigcup_{t \geq 0} tC$

Examples

Let $X = \mathbb{R}^n$.

- Any affine space is convex.
- Any hyperplane of X can be defined as $H := \{x \in X \mid \ a^\top x = b\}$ for some $a \in \mathbb{R}^n$ and $b \in \mathbb{R}$ and is an affine space of dimension $n-1$.
- H divide X into two half-spaces $\{x \in \mathbb{R}^n \mid a^\top x \leq b\}$ and $\{x \in \mathbb{R}^n \mid a^\top x \geq b\}$ which are (closed) convex sets.
- For any norm $\| \cdot \|$ the ball $B_{\| \cdot \|}(x_0, r) := \{x \in X \mid \|x - x_0\| \leq r\}$ is a (closed) convex set.
  ♣ Exercise: Prove it.
- The set $C = \{(x, t) \in X \times \mathbb{R} \mid \|x\| \leq t \ \}$ is a cone.
- The set $C = \{x \in X \mid Ax \leq b\}$ where A and b are given is a (closed) convex set called polyhedron and polytope if it is bounded.

Operations preserving convexity ♡

Assume that all sets denoted by C (indexed or not) are convex.

- $C_1 + C_2$ and $C_1 \times C_2$ are convex sets.
- For any arbitrary index set $\mathcal{I}$ the intersection $\bigcap_{i \in \mathcal{I}} C_i$ is convex.
- Let f be an affine function. Then $f(C)$ and $f^{-1}(C)$ are convex.
- In particular, $C + x_0$, and tC are convex. The projection of C on any affine space is convex.
- The closure $\mathrm{cl}(C)$ and relative interior $\mathrm{ri}(C)$ are convex.

♣ Exercise: Prove these results.

Perspective and linear-fractional function ♢

Let $P : \mathbb{R}^n \times \mathbb{R} \to \mathbb{R}^n$ be the perspective function defined as $P(x, t) = x/t$, with $\mathrm{dom}(P) = \mathbb{R}^n \times \mathbb{R}_+^*$.

Theorem

If $C \subset \mathrm{dom}(P)$ is convex, then $P(C)$ is convex.
If $C \subset \mathbb{R}^n$ is convex, then $P^{-1}(C)$ is convex.

♠ Exercise: Prove this result.

Let $f : \mathbb{R}^n \to \mathbb{R}^m$ be a linear-fractional function of the form $f(x) := (Ax + b)/(c^\top x + d)$, with $\mathrm{dom}(f) = \{x \mid c^\top x + d > 0\}$.

Theorem

If $C \subset \mathrm{dom}(f)$ is convex, then $f(C)$ and $f^{-1}(C)$ are convex.

♣ Exercise: prove this result.

Cone ordering

Let $K \subset \mathbb{R}^n$ be a closed, convex, pointed cone with non-empty interior. We define the cone ordering according to K by

$$x \preceq_K y \iff y - x \in K.$$

Examples:

- For $K = \mathbb{R}^n_+$, $\preceq_K$ is the componentwise ordering, *i.e.*,

$$x \preceq_{\mathbb{R}^n_+} y \iff \forall i \in [n],\ x_i \leq y_i.$$

- For $K = S_n^+(\mathbb{R})$, $\preceq_K$ is the Löwner ordering, *i.e.*,

$$A \preceq_{S_n^+} B \iff B - A \in S_n^+(\mathbb{R}).$$

♣ Exercise: Prove that $\preceq_K$ is a partial order (*i.e.*,reflexive, antisymmetric, transitive) compatible with scalar product, addition and limits.

Separation ♢

Let X be a Banach space, and X^* its topological dual (i.e. the set of all continuous linear forms on X).

Theorem (Simple separation)

Let A and B be convex non-empty, disjoint subsets of X. There exists a separating hyperplane $(x^*, \alpha) \in X^* \times \mathbb{R}$ such that

$$\langle x^*, a \rangle \leq \alpha \leq \langle x^*, b \rangle \qquad \forall a, b \in A \times B.$$

Theorem (Strong separation)

Let A and B be convex non-empty, disjoint subsets of X. Assume that, A is closed, and B is compact (e.g. a point), then there exists a strict separating hyperplane $(x^*, \alpha) \in X^* \times \mathbb{R}$ such that, there exists $\varepsilon > 0$,

$$\langle x^*, a \rangle + \varepsilon \leq \alpha \leq \langle x^*, b \rangle - \varepsilon \qquad \forall a, b \in A \times B.$$

Remark: In Banach spaces these theorems require the Zorn Lemma which is equivalent to the axiom of choice. We will stay in finite dimension where these theorems are easier to prove.

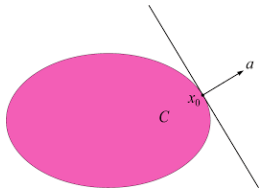
Supporting hyperplane ♢

Theorem

Let $x_0 \notin \mathrm{ri}(C)$ and C convex. Then there exists $a \neq 0$ such that

$$a^\top x \geq a^\top x_0, \qquad \forall x \in C$$

If $x_0 \in C$, say that $H = \{x \mid a^\top x = a^\top x_0\}$ is a supporting hyperplane of C at x_0.



♣ Exercise: prove this theorem
Remark: there can be more than one supporting hyperplane at a given point.

Convex set as intersection of half-spaces ♢

- The closed convex hull of $C \subset X$, denoted $\overline{\mathrm{conv}}(C)$ is the smallest closed convex set containing C.
- $\overline{\mathrm{conv}}(C)$ is the intersection of all the half-spaces containing C.
- A polyhedron is a finite intersection of half-spaces while a convex set is a possibly non-finite intersection of half-spaces.

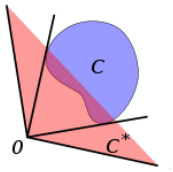
Dual and normal cones

- Let $C \subset \mathbb{R}^n$ be a set. We define its (positive) dual cone by

$$C^{\oplus} := \{x \mid x^\top c \geq 0, \quad \forall c \in C\}$$

- For any set C, $C^{\oplus}$ is a closed convex cone.
- The normal cone of C at x_0 is

$$N_C(x_0) := \{\lambda \in \mathbb{R}^n \mid \lambda^\top (x - x_0) \leq 0, \forall x \in C\}$$



Let $K \subset \mathbb{R}^n$ be a cone.

- $K^{\oplus}$ is closed convex.
- $K_1 \subset K_2$ implies $K_2^{\oplus} \subset K_1^{\oplus}$
- $K^{\oplus\oplus} = \overline{\text{conv}}\, K$

♣ Exercise: Prove these results

Examples

- The positive orthant $K = \mathbb{R}^n_+$ is a self dual cone, that is $K^{\oplus} = K$.
- In the space of symmetric matrices $S_n(\mathbb{R})$, with the scalar product $\langle A, B \rangle = \text{tr}(AB)$, the set of positive semidefinite matrices $K = S_n^+(\mathbb{R})$ is self dual.
- Let $\|\cdot\|$ be a norm. The cone $K = \{(x, t) \mid \|x\| \leq t\}$ has for dual $K^{\oplus} = \{(\lambda, z) \mid \|\lambda\|_\star \leq z\}$, where $\|\lambda\|_\star := \sup_{x:\|x\|\leq 1} \lambda^\top x$.

♠ Exercise: prove these results

Functions with non finite values ♡

- It is very useful in optimization to allow functions to take non-finite values, that is to take values in $\bar{\mathbb{R}} := \mathbb{R} \cup \{-\infty, +\infty\}$.
- If both $-\infty$ and $+\infty$ are allowed be very careful of each addition[1]!
- Let $f : X \to \bar{\mathbb{R}}$. We define
  - The epigraph of f as
$$\text{epi}(f) := \{(x, t) \in X \times \mathbb{R} \quad | \quad f(x) \leq t\}$$
  - the domain of f as
$$\text{dom}(f) := \{x \in X \quad | \quad f(x) < +\infty\}.$$
  - The sublevel set of level α
$$lev_\alpha(f) := \{x \in X \quad | \quad f(x) \leq \alpha\}.$$
- f is said to be lower semicontinuous (l.s.c.) if $\text{epi}(f)$ is closed.
- f is said to be proper if it never takes value $-\infty$, has a non-empty domain (at least one finite value).

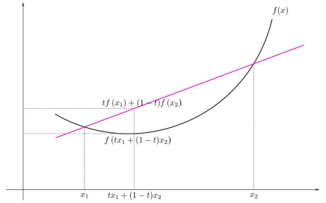
[1] Which is why we usually consider proper functions.

Convex function ♡

- A function $f : X \to \bar{\mathbb{R}}$ is convex if its epigraph is convex.
- $f : X \to \mathbb{R} \cup \{+\infty\}$ is convex iff

$$\forall t \in [0, 1], \forall x, y \in X, \quad f(tx + (1-t)y) \leq tf(x) + (1-t)f(y)$$

- f is concave if $-f$ is convex.



Basic properties ♡

- If f, g convex, $t > 0$, then $tf + g$ is convex.
- If f convex non-decreasing, g convex, then $f \circ g$ convex.
- If f convex and a affine, then $f \circ a$ is convex.
- If $(f_i)_{i \in I}$ is a family of convex functions, then $\sup_{i \in I} f_i$ is convex.
- The domain and the sublevel sets of a convex function are convex.
- A convex function is always above its tangents (supporting hyperplanes).

♣ Exercise: Prove these results.

Theorem (Jensen inequality)

Let f be a convex function and $\boldsymbol{X}$ an integrable random variable. Then we have

$$f(\mathbb{E}[\boldsymbol{X}]) \leq \mathbb{E}[f(\boldsymbol{X})].$$

Convex function: regularity ♡

Consider a convex function $f : \mathbb{R}^n \to \mathbb{R} \cup \{+\infty\}$.

- f is continuous (on $\mathbb{R}^n$) if and only if $\mathrm{dom}(f) = \mathbb{R}^n$ (i.e., if it is finite everywhere).
- f is continuous on the interior of its domain.
- f is Lipschitz on any compact convex set contained in the interior of its domain.

Convex functions: strict and strong convexity ♡

- $f : X \to \mathbb{R} \cup \{+\infty\}$ is strictly convex iff
$$\forall t \in]0,1[, \quad \forall x, y \in X, \qquad f(tx + (1-t)y) < tf(x) + (1-t)f(y)$$
- $f : X \to \mathbb{R} \cup \{+\infty\}$ is α-strongly convex iff $\forall t \in]0,1[, \quad \forall x, y \in X$,
$$f(tx + (1-t)y) \leq tf(x) + (1-t)f(y) - \frac{1}{2}\alpha t(1-t)\|x - y\|^2$$
- If $f \in C^1(\mathbb{R}^n)$
  - $\langle \nabla f(x) - \nabla f(y), x - y \rangle \geq 0$ iff f convex
  - if strict inequality holds, then f strictly convex
  - $f : X \to \mathbb{R} \cup \{+\infty\}$ is α-strongly convex iff $\forall x, y \in X$
$$f(y) \geq f(x) + \langle \nabla f(x), y - x \rangle + \frac{\alpha}{2}\|y - x\|^2$$
- If $f \in C^2(\mathbb{R}^n)$,
  - $\nabla^2 f \succcurlyeq 0$ iff f convex
  - if $\nabla^2 f \succ 0$ then f strictly convex
  - if $\nabla^2 f \succcurlyeq \alpha I$ then f is α-strongly convex

Three convexity tests you will actually use ♡

Test 1 — Definition (segment / Jensen)

To prove f convex on a convex domain $\mathrm{dom}(f)$, show

$$\forall x, y \in \mathrm{dom}(f), \ \forall t \in [0,1], \qquad f(tx + (1-t)y) \leq tf(x) + (1-t)f(y).$$

Test 2 — Epigraph / sublevel sets (geometric)

f is convex $\iff$ its epigraph is convex:

$$\mathrm{epi}(f) = \{(x, s) \in \mathbb{R}^n \times \mathbb{R} \mid f(x) \leq s\} \text{ is convex.}$$

Test 3 — Smooth test (first/second order)

If $f \in C^1$ on an open convex set:

$$f \text{ convex} \iff \langle \nabla f(x) - \nabla f(y), x - y \rangle \geq 0 \quad \forall x, y.$$

If $f \in C^2$:

$$f \text{ convex} \iff \nabla^2 f(x) \succeq 0 \ \ \forall x, \qquad f \ \alpha\text{-strongly convex} \iff \nabla^2 f(x) \succeq \alpha I.$$

Important examples

- The indicator function of a set $C \subset X$,

$$\mathbb{I}_C(x) := \begin{cases} 0 & \text{if } x \in C \\ +\infty & \textit{otherwise} \end{cases}$$

  is convex iff C is convex.
- $x \mapsto e^{ax}$ is convex for any $a \in \mathbb{R}$
- $x \mapsto \|x\|^q$ is convex for $q \geq 1$ and any norm
- $x \mapsto \ln(x)$ is concave
- $x \mapsto x\ln(x)$ is convex
- $x \mapsto \ln(\sum_{i=1}^n e^{x_i})$ is convex

Convex optimization problem ♡

$$\underset{x \in C}{\text{Min}} \quad f(x)$$

Where C is non-empty, closed convex and f convex finite valued[2], is a convex optimization problem.

- If C is compact and f proper lsc, then there exists an optimal solution.
- If f is proper lsc and coercive, then there exists an optimal solution.
- The set of optimal solutions is convex.
- If f is strictly convex the minimum (if it exists) is unique.
- If f is α-strongly convex (and proper lsc) the minimum exists and is unique.

♣ Exercise: Prove these results.

[2] Alternatively, f can be proper lower semicontinuous. Here I assumed $C \subset \text{dom}(f)$.

Optimality conditions ♡

Note that minimizing f over C or minimizing $f + \mathbb{I}_C$ over X is the same thing.
We consider the (unconstrained) optimization problem

$$\underset{x \in X}{\text{Min}} \quad f(x),$$

with $x^\sharp$ an optimal solution and f not necessarily convex.

- If f is differentiable, then $\nabla f(x^\sharp) = 0$.
- If f is twice differentiable, then $\nabla^2 f(x^\sharp) \succeq 0$.
- If f is twice differentiable and $\nabla f(x_0) = 0$ and $\nabla^2 f(x_0) \succ 0$ then x_0 is a local minimum.

If, in addition, f is convex then $\nabla f(x) = 0$ is a necessary and sufficient optimality condition.

Partial infimum ♡

Let f be a convex function and C a convex set. The function

$$g : x \mapsto \inf_{y \in C} f(x, y)$$

is convex.

♠ Exercise: Prove this result.

♣ Exercise: Prove that the function distance to a convex set C defined by

$$d_C(x) := \inf_{c \in C} \|c - x\|$$

is convex.

Perspective function

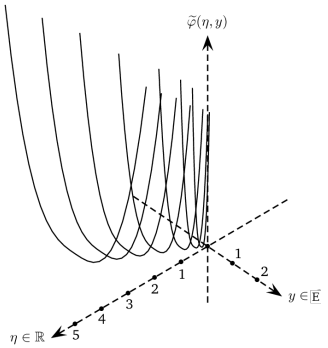
Let $\phi : E \to \overline{\mathbb{R}}$. The perspective of ϕ is defined as $\tilde{\phi} : \mathbb{R}_+^* \times E \to \mathbb{R}$ by

$$\tilde{\phi}(\eta, y) := \eta\phi(y/\eta).$$

Theorem

ϕ is convex iff $\tilde{\phi}$ is convex.

♠ Exercise: prove this result



Inf-Convolution

Let f and g be proper function from X to $\mathbb{R} \cup \{+\infty\}$. We define

$$f \square g : x \mapsto \inf_{y \in X} f(y) + g(x - y)$$

♣ Exercise: Show that

- $f \square g = g \square f$
- If f and g are convex then so is $f \square g$

Subdifferential of convex function

Let X be an Hilbert space, $f : X \to \bar{\mathbb{R}}$ convex.

- The subdifferential of f at $x \in \mathrm{dom}(f)$ is the set of slopes of all affine minorants of f exact at x:
$$\partial f(x) := \left\{ \lambda \in X \quad | \quad f(\cdot) \geq \langle \lambda, \cdot - x \rangle + f(x) \right\}.$$
- If f is differentiable at x then
$$\partial f(x) = \{\nabla f(x)\}.$$

Examples

- If $f : x \mapsto |x|$, then
$$\partial f(x) = \begin{cases} -1 & \text{if } x < 0 \\ [-1, 1] & \text{if } x = 0 \\ 1 & \text{if } x > 0 \end{cases}$$
- If C is convex then, for $x \in C$, $\partial(\mathbb{I}_C)(x) = N_C(x)$
♣ Exercise: Prove it.
- If f_1 and f_2 are convex and differentiable. Define $f = \max(f_1, f_2)$. Then
  - if $f_1(x) > f_2(x)$, $\partial f(x) = \{\nabla f_1(x)\}$
  - if $f_1(x) < f_2(x)$, $\partial f(x) = \{\nabla f_2(x)\}$;
  - if $f_1(x) = f_2(x)$, $\partial f(x) = \overline{\mathrm{conv}}(\{\nabla f_1(x), \nabla f_2(x)\})$.

Subdifferential calculus

Let f_1 and f_2 be proper convex functions.

Theorem (Moreau-Rockafellar)

We have

$$\partial(f_1)(x) + \partial(f_2)(x) \subset \partial(f_1 + f_2)(x), \qquad \forall x$$

Further if $\mathrm{ri}(\mathrm{dom}(f_1)) \cap \mathrm{ri}(\mathrm{dom}(f_2)) \neq \emptyset$ *then*

$$\partial(f_1)(x) + \partial(f_2)(x) = \partial(f_1 + f_2)(x), \qquad \forall x$$

When f_i *is polyhedral you can replace* $\mathrm{ri}(\mathrm{dom}(f_i))$ *by* $\mathrm{dom}(f_i)$ *in the condition.*

Theorem

If f *is convex and* $a : x \mapsto Ax + b$ *with* $Im(a) \cap \mathrm{ri}(\mathrm{dom}(f)) \neq \emptyset$, *then*

$$\partial(f \circ a)(x) = A^\top \partial f(Ax + b).$$

Counterexample to the qualification in Moreau–Rockafellar

$$f(x) = \begin{cases} -\sqrt{x}, & x \geq 0, \\ +\infty, & x < 0, \end{cases} \qquad g(x) = f(-x).$$

Then $\mathrm{dom}\, f = [0, \infty)$, $\mathrm{dom}\, g = (-\infty, 0]$, hence $\mathrm{dom}\, f \cap \mathrm{dom}\, g = \{0\}$ and

$$(f + g)(x) = \begin{cases} 0, & x = 0, \\ +\infty, & x \neq 0, \end{cases} \quad \Rightarrow \quad f + g = \delta_{\{0\}}.$$

Therefore $\partial(f + g)(0) = \mathbb{R}$. On the other hand, $\partial f(0) = \partial g(0) = \emptyset$, so

$$\partial f(0) + \partial g(0) = \emptyset \subsetneq \mathbb{R} = \partial(f + g)(0).$$

First order optimality conditions

Theorem

Let $f : X \mapsto \mathbb{R} \cup \{+\infty\}$ *be a convex function (not necessarily) differentiable.* $x^\sharp$ *is a minimizer of* f *if and only if* $0 \in \partial f(x^\sharp)$.

Theorem

Let f *be a proper convex function and* C *a closed non-empty convex set such that* $\mathrm{ri}(C) \cap \mathrm{ri}(\mathrm{dom}(f)) \neq \emptyset$ *then* $x^\sharp$ *is an optimal solution to*

$$\min_{x \in C} \quad f(x)$$

iff

$$0 \in \partial f(x^\sharp) + N_C(x^\sharp),$$

iff

$$\exists \lambda \in \partial f(x^\sharp), \quad \lambda \in -N_C(x^\sharp).$$

Normal cone, Tangent cone and optimality

Let C be a convex set. We define the tangent cone of $C \subset \mathbb{R}^n$ at point $x \in C$, as the set of directions in which you can move from x while staying in C for some time, that is

$$T_C(x) := \overline{\left\{ \lambda(y - x) \;\middle|\; y \in C, \quad \lambda \in \mathbb{R}^+ \right\}}$$

In particular, $T_C(x) = \mathbb{R}^n$ iff $x \in int(C)$.

♣ Exercise: Prove that $[T_C(x)]^\oplus = -N_C(x)$.

Partial infimum

Let $f : X \times Y \to \bar{\mathbb{R}}$ be a jointly convex and proper function, and define

$$v(x) = \inf_{y \in Y} f(x, y)$$

then v is convex.
If v is proper, and $v(x) = f(x, y^\sharp(x))$ then

$$\partial v(x) = \{g \in X \quad | \quad (g, 0) \in \partial f(x, y^\sharp(x))\}$$

proof:

$$\begin{aligned} g \in \partial v(x) &\Leftrightarrow \forall x', \quad v(x') \geq v(x) + \langle g, x' - x \rangle \\ &\Leftrightarrow \forall x', y' \quad f(x', y') \geq f(x, y^\sharp(x)) + \left\langle \begin{pmatrix} g \\ 0 \end{pmatrix}, \begin{pmatrix} x' \\ y' \end{pmatrix} - \begin{pmatrix} x \\ y^\sharp(x) \end{pmatrix} \right\rangle \\ &\Leftrightarrow \begin{pmatrix} g \\ 0 \end{pmatrix} \in \partial f(x, y^\sharp(x)) \end{aligned}$$

Convex function: regularity

- Assume f convex, then f is continuous on the relative interior of its domain, and Lipschitz on any compact contained in the relative interior of its domain.
- A proper convex function is subdifferentiable on the relative interior of its domain.
- If f is convex, it is L-Lipschitz iff $\partial f(x) \subset B(0, L), \quad \forall x \in \mathrm{dom}(f)$

Fenchel transform

Let X be a Hilbert space, $f : X \to \bar{\mathbb{R}}$ be a proper function.

- The Fenchel transform of f, is $f^\star : X \to \bar{\mathbb{R}}$ with

$$f^\star(\lambda) := \sup_{x \in X} \langle \lambda, x \rangle - f(x).$$

- $f^\star$ is convex lsc as the supremum of affine functions.
- $f \leq g$ implies that $f^\star \geq g^\star$.
- If f is proper convex lsc, then $f^{\star\star} = f$, otherwise $f^{\star\star} \leq f$.

♣ Exercise: Prove the first two points

Fenchel transform and subdifferential

- By definition $f^\star(\lambda) \geq \langle \lambda, x \rangle - f(x)$ for all x,
- thus we always have (Fenchel-Young) $f(x) + f^\star(\lambda) \geq \langle \lambda, x \rangle$.
- Recall that $\lambda \in \partial f(x)$ iff,

$$f(x') \geq f(x) + \langle \lambda, x' - x \rangle, \qquad \forall x'$$

iff

$$\langle \lambda, x \rangle - f(x) \geq \langle \lambda, x' \rangle - f(x') \qquad \forall x'$$

that is

$$\lambda \in \partial f(x) \Leftrightarrow x \in \arg\max_{x' \in X} \{\langle \lambda, x' \rangle - f(x')\} \Leftrightarrow f(x) + f^\star(\lambda) = \langle \lambda, x \rangle$$

- If f proper convex lsc

$$\lambda \in \partial f(x) \iff x \in \partial f^\star(\lambda).$$

What you have to know

- What is a affine set, a convex set, a polyhedron, a (convex) cone
- What is a convex function, that it is above its tangents.
- Jensen inequality
- What is a convex optimization problem. That any local minimum is a global minimum.
- The necessary optimality condition $\nabla f(x^\sharp) \in [T_X(x^\sharp)]^\oplus$

What you really should know

- That you can separate convex sets with a linear function
- What is the positive dual of a cone
- Basic manipulations preserving convexity (sum, cartesian product, intersection, linear projection)
- What is the domain, the sublevel of a function f
- What is a lower semicontinuous function, a proper convex function
- Conditions of (strict, strong) convexity for differentiable functions
- The partial minimum of a convex function is convex
- The definition of the subdifferential.
- The definition of the Fenchel transform.
- The link between Fenchel transform and subdifferential.

What you have to be able to do

- Show that a set is convex
- Show that a function is (strictly, strongly) convex
- Go from constrained problem to unconstrained problem using the indicator function $\mathbb{I}_X$

What you should be able to do

- Compute dual cones
- Use advanced results (projection, partial infimum, perspective) to show that a function or a set is convex
- Compute the Fenchel transform of simple functions

Optimality conditions

V. Leclère (ENPC)

March 13th, 2026

Why should I bother to learn this stuff?

- Optimality conditions enable to solve exactly some easy optimization problems (e.g. in microeconomics, some mechanical problems...)
- Optimality conditions are used to derive algorithms for complex problems
- KKT is a certificate of optimality (or non optimality) used by optimization solvers (more than a solving method per se).
- $\Longrightarrow$ fundamental both for studying optimization as well as other science

Optimization problem: vocabulary ♡

Generically speaking, an optimization problem is

$$\min_{x \in X} \quad f(x) \qquad (P)$$

where

- $f : \mathbb{R}^n \to \mathbb{R}$ is the objective function (a.k.a. cost function),
- X is the feasible set,
- $x \in X$ is an admissible decision variable or a solution,
- $x^\sharp \in X$ such that $val(P) = f(x^\sharp) = \inf_{x \in X} f(x)$ is an optimal solution,
- if $X = \mathbb{R}^n$ the problem is unconstrained,
- if X and f are convex, then the problem is convex,
- if X is a polyhedron and f linear then the problem is linear,
- if X is a convex cone (optionally intersected with affine space) and f linear then the problem is conic.

Optimization problem: explicit formulation ♡

The previous optimization problem is often defined explicitly in the following standard form

$$\begin{aligned} \min_{x \in \mathbb{R}^n} \quad & f(x) && (P) \\ \text{s.t.} \quad & g_i(x) = 0 && \forall i \in [n_E] \\ & h_j(x) \leq 0 && \forall j \in [n_I] \end{aligned}$$

with

$$X := \{x \in \mathbb{R}^n \mid \forall i \in [n_E], \quad g_i(x) = 0, \quad \forall j \in [n_I], \quad h_j(x) \leq 0\}.$$

- (P) is a differentiable optimization problem if f and $\{g_i\}_{i \in [n_E]}$ and $\{h_j\}_{j \in [n_I]}$ are differentiable.
- (P) is a convex differentiable optimization problem if f, and h_j (for $j \in [n_I]$) are convex differentiable and g_i (for $i \in [n_E]$) are affine.
  ♣ Exercise: Show that in this case X is convex.

A few remarks and tricks ♢

- We can always write an abstract optimization problem in standard form (exercise!)
- For a given optimization problem there is an infinite number of possible standard forms (exercise!)
- We can always find an equivalent problem in dimension $\mathbb{R}^{n+1}$ with linear cost (exercise!)
- A minimization problem with $X = \emptyset$ has value $+\infty$ (by convention)
- A minimization problem has value $-\infty$ iff there exists a sequence $x_n \in X$ such that $f(x_n) \to -\infty$
- Maximizing f is just minimizing $-f$

Differentiable case ♡

Theorem

Assume that $f : \mathbb{R}^n \to \bar{\mathbb{R}}$ is differentiable at $x^\sharp$.

1. *If $x^\sharp$ is an unconstrained local minimizer of f then $\nabla f(x^\sharp) = 0$.*
2. *If in addition f is convex, then $\nabla f(x^\sharp) = 0$ iff $x^\sharp$ is a global minimizer.*

Proof:

1. Assume $\nabla f(x^\sharp) \neq 0$. DL of order 1 at $x^\sharp$ show that $f(x^\sharp - t\nabla f(x^\sharp)) < f(x^\sharp)$ for $t > 0$ small enough.
2. $f(y) \geq f(x^\sharp) + \langle \nabla f(x^\sharp), y - x^\sharp \rangle$.

Convex case ♡

Theorem

Consider $f : \mathbb{R}^n \to \bar{\mathbb{R}}$. Then $x^\sharp$ is a global minimum iff

$$0 \in \partial f(x^\sharp)$$

Theorem

Consider a proper convex function $f : \mathbb{R}^n \to \bar{\mathbb{R}}$, and X a closed convex set, such that $\mathrm{ri}(\mathrm{dom}(f)) \cap \mathrm{ri}(X) \neq \emptyset$.
Then $x^\sharp$ is a minimizer of f on X iff there exists $g \in \partial f(x^\sharp)$ such that $-g \in N_X(x^\sharp)$.

proof : The technical assumption ensures that $\partial(f + \mathbb{I}_X) = \partial f + \partial(\mathbb{I}_X)$.
As $\partial(\mathbb{I}_X) = N_X$, we have, $0 \in \partial(f + \mathbb{I}_X)(x^\sharp)$ iff there exists $g \in \partial f(x^\sharp)$ such that $-g \in N_X(x^\sharp)$.

From local minimality to a tangent-direction condition ♡

Consider

$$(P) \qquad \min_{x \in X} f(x),$$

where $f : \mathbb{R}^n \to \mathbb{R}$ is differentiable and $X \subset \mathbb{R}^n$ is closed. Let $x \in X$ be a local minimizer. Take any sequence of unit vectors $d_k \to d$ such that

$$x + \tfrac{1}{k} d_k \in X \qquad \text{for } k \text{ large enough.}$$

Then $x + \frac{1}{k} d_k \to x$, hence by local minimality,

$$f(x + \tfrac{1}{k} d_k) \;\geq\; f(x) \qquad \text{for } k \text{ large enough.}$$

Since f is differentiable at x,

$$f(x + \tfrac{1}{k} d_k) = f(x) + \tfrac{1}{k} \langle \nabla f(x), d_k \rangle + o(\tfrac{1}{k}).$$

Letting $k \to \infty$ (so $k\, o(1/k) \to 0$ and $d_k \to d$) yields

$$\langle \nabla f(x), d \rangle \geq 0.$$

Tangent cones

For $f : \mathbb{R}^n \to \mathbb{R}$, we consider an optimization problem of the form

$$\min_{x \in X} \quad f(x).$$

Definition (Bouligand cone)

We say that $d \in \mathbb{R}^n$ is tangent to X at $x \in X$ if there exists a sequence $x_k \in X$ converging to x and a sequence $t_k \searrow 0$ such that

$$d = \lim_k \frac{x_k - x}{t_k}.$$

Let $T_X(x)$ be the tangent cone of X at x, that is, the set of all tangent vectors to X at x.

Equivalently,

$$T_X(x) = \{\ d \in \mathbb{R}^n \ \mid \exists t_k \searrow 0,\ \exists d_k \to d,\ x + t_k d_k \in X\}$$

Optimality conditions - differentiable case

Consider a function $f : \mathbb{R}^n \to \mathbb{R}$ and the optimization problem

$$(P) \qquad \min_{x \in X} \quad f(x).$$

If $x^\sharp \notin \operatorname{int}(X)$ we do not necessarily need to have $\nabla f(x^\sharp) = 0$, indeed we just to have $\langle d, \nabla f(x^\sharp) \rangle \geq 0$ for all "admissible" direction d.

Theorem

Assume that f is differentiable at $x^\sharp$.

1. *If $x^\sharp$ is a local minimizer of (P) we have*

$$\nabla f(x^\sharp) \in \left[T_X(x^\sharp)\right]^\oplus. \qquad (*)$$

2. *If f and X are both convex, and $(*)$ holds, then $x^\sharp$ is an optimal solution of (P)*

♠ Exercise: Prove this result.

Convex case

Let $K_X^{ad}(x)$ be the cone of admissible direction

$$K_X^{ad}(x) := \left\{t(y - x) \in \mathbb{R}^n \ \mid \ y \in X, \quad t \geq 0\right\}$$

Lemma

If $X \subset \mathbb{R}^n$ is convex, and $x \in X$, we have

$$T_X(x) = \overline{K_X^{ad}(x)}.$$

Recall that

$$T_X(x) = \{\ d \in \mathbb{R}^n \ \mid \exists t_k \searrow 0,\ \exists d_k \to d,\ x + t_k d_k \in X\}$$

♠ Exercise: Prove this lemma

Differentiable constraints

We consider the following set of admissible solution

$$X = \left\{x \in \mathbb{R}^n \ \mid \ g_i(x) = 0,\ i \in [n_E] \quad h_j(x) \leq 0,\ j \in [n_I]\right\},$$

where g and h are differentiable functions.

Recall that the tangent cone is given by

$$T_X(x) = \{\ d \in \mathbb{R}^n \ \mid \exists t_k \searrow 0,\ \exists d_k \to d,\ g(x + t_k d_k) = 0,\ h(x + t_k d_k) \leq 0\}$$

We define the linearized tangent cone

$$T_X^\ell(x) := \{\ d \in \mathbb{R}^n \ \mid \left\langle \nabla g_i(x)\,, d \right\rangle = 0,\ \forall i \in [n_E] \\ \left\langle \nabla h_j(x)\,, d \right\rangle \leq 0,\ \forall j \in I_0(x)\}$$

where

$$I_0(x) := \left\{j \in [n_I] \ \mid \ h_j(x) = 0\right\}.$$

Constraint qualifications ♢

We always have

$$T_X(x) \subset T_X^\ell(x).$$

♣ Exercise: Prove it.
We say that the constraints are qualified[1] at x if

$$T_X(x) = T_X^\ell(x).$$

Constraint qualifications ensure the existence of Lagrange multipliers at local minima.

[1] This is Abadie CQ, there is a weaker version called Guignard CQ: $(T_X(x)^\oplus = (T_X^\ell(x))^\oplus$.

Sufficient qualification conditions ♡

Recall that g and h are assumed differentiable.
We denote the index set of active constraints at x

$$I_0(x) := \{j \in [n_I] \quad | \quad h_j(x) = 0\}.$$

The following conditions are sufficient qualification conditions at x:

1. g and h_i for $i \in I_0(x)$ are affine in a neighborhood of x;
2. (Slater) g is affine, h_j are convex, and there exists x_S such that $g(x_S) = 0$ and $h_j(x_S) < 0$;
3. (Mangasarian-Fromowitz) For all $\alpha \in \mathbb{R}^{n_E}$ and $\beta \in \mathbb{R}_+^{n_{I_0(x)}}$,

$$\sum_{i\in[n_E]} \alpha_i \nabla g_i(x) + \sum_{j\in I_0(x)} \beta_j \nabla h_j(x) = 0 \qquad \Longrightarrow \qquad \alpha = 0 \text{ and } \beta = 0$$

Expliciting the optimality condition I ♢

Under constraint qualification, the optimality condition reads

$$\nabla f(x) \in \left[T_X^\ell(x)\right]^\oplus$$

where

$$T_X^\ell(x) = \{\ d \in \mathbb{R}^n \ \mid \underbrace{\left\langle \nabla g_i(x)\,, d \right\rangle = 0, i \in [n_E] \quad \left\langle \nabla h_j(x)\,, d \right\rangle \le 0, j \in I_0(x)}_{= A_x d \in C}\}.$$

with $A_x = \begin{pmatrix} ((\nabla g_i(x))^\top)_{i\in[n_E]} \\ ((\nabla h_j(x))^\top)_{j\in I_0(x)} \end{pmatrix}$ and $C = \{0\}^{n_E} \times (\mathbb{R}_-)^{n_I}$.
♣ Exercise: Show that $C^\oplus = \mathbb{R}^{n_E} \times (\mathbb{R}_-)^{|I_0(x)|}$

Expliciting the optimality condition II ♢

Recall that the positive dual cone of a set K is

$$K^\oplus := \{d \in \mathbb{R}^n \quad | \quad \langle d, x\rangle \ge 0, \forall x \in K\}.$$

Let C be a closed convex cone. Consider

$$K = A^{-1}C := \{x \in \mathbb{R}^n \mid Ax \in C\ \},$$

then

$$K^\oplus = \{\ A^\top \lambda \mid \lambda \in C^\oplus\}.$$

♣ Exercise: prove it.
Hence,

$$\nabla f(x) \in \big[\underbrace{T_X^\ell(x)}_{A_x^{-1}C}\big]^\oplus$$

$$\iff \qquad \exists \lambda \in C^\oplus, \quad \nabla f(x) + A_x^\top \lambda = 0$$

$$\iff \quad \exists \lambda \in \mathbb{R}^{n_E},\ \exists \mu \in \mathbb{R}_+^{I_0(x)} \quad \nabla f(x) + \sum_{}^{n_E} \lambda_i \nabla g_i(x) + \sum_{} \mu_i \nabla h_i(x) = 0.$$

Karush Kuhn Tucker condition ♡

Theorem (KKT)

Assume that the objective function f and the constraint function g_i and h_j are differentiable. Assume that the constraints are qualified at x.

Then if x is a local minimum of

$$\min_{\tilde{x}\in\mathbb{R}^n} \left\{ f(\tilde{x}) \quad | \quad g_i(\tilde{x}) = 0,\ \forall i \in [n_E] \quad h_j(\tilde{x}) \leq 0,\ \forall j \in [n_I] \right\}$$

then there exists dual variables λ, μ such that

$$(KKT) \quad \begin{cases} \nabla f(x) + \sum_{i=1}^{n_E} \lambda_i \nabla g_i(x) + \sum_{j=1}^{n_I} \mu_j \nabla h_j(x) = 0 & \nabla_x \mathcal{L} = 0 \\ g(x) = 0, \quad h(x) \leq 0 & \text{Primal feasibility} \\ \lambda \in \mathbb{R}^{n_E}, \quad \mu \in \mathbb{R}^{n_I}_+ & \text{dual feasibility} \\ \mu_j h_j(x) = 0 \qquad \forall j \in [n_I] & \text{complementarity constraint} \end{cases}$$

KKT conditions in the convex case ♡

Theorem (KKT is necessary and sufficient in convex case)

Assume that f and h_i are convex differentiable functions, and that g_j are affine functions. Moreover, assume that the Slater's condition holds: $\exists x_S \in \mathbb{R}^n$ such that

$$g_i(x_S) = 0\ \forall i \in [n_E], \qquad h_j(x_S) < 0\ \forall j \in [n_I].$$

Then x is a (global) minimum of

$$\min_{\tilde{x}\in\mathbb{R}^n} \left\{ f(\tilde{x}) \mid g_i(\tilde{x}) = 0\ \forall i \in [n_E], \quad h_j(\tilde{x}) \leq 0\ \forall j \in [n_I] \right\}$$

if and only if there exist dual variables λ, μ such that (x, λ, μ) satisfy (KKT).

What you have to know

- Basic vocabulary: objective, constraint, admissible solution, differentiable optimization problem
- First order necessary KKT conditions

What you really should know

- What is a tangent cone
- Sufficient qualification conditions (linear and Slater's)
- That KKT conditions are sufficient in the convex case

What you have to be able to do

- Write the KKT condition for a given explicit problem and use them to solve said problem

What you should be able to do

- Check that constraints are qualified

Duality

V. Leclère (ENPC)

March 20th, 2026

Why should I bother to learn this stuff?

- Duality allows a second representation of the same convex problem, giving sometimes some interesting insights (e.g. principle of virtual forces in mechanics)
- Duality is a good way of obtaining lower bounds
- Duality is a powerful tool for decomposition methods
- $\Longrightarrow$ fundamental both for studying optimization (continuous and operations research)
- $\Longrightarrow$ useful in other fields like mechanics and machine learning

Min-Max duality ♡

Consider the following problem

$$\operatorname*{Min}_{x\in\mathcal{X}} \sup_{y\in\mathcal{Y}} \Phi(x,y)$$

where, for the moment, $\mathcal{X}$ and $\mathcal{Y}$ are arbitrary sets, and Φ an arbitrary function.
By definition the dual of this problem is

$$\operatorname*{Max}_{y\in\mathcal{Y}} \inf_{x\in\mathcal{X}} \Phi(x,y)$$

and we have weak duality, that is

$$\sup_{y\in\mathcal{Y}} \inf_{x\in\mathcal{X}} \Phi(x,y) \leq \inf_{x\in\mathcal{X}} \sup_{y\in\mathcal{Y}} \Phi(x,y)$$

♣ Exercise: Prove this result. Getting equality is more tricky...

Dual representation of some characteristic functions

Recall that, if $X \subset \mathbb{R}^n$

$$\mathbb{I}_X(x) = \begin{cases} 0 & \text{if } x \in X \\ +\infty & \text{otherwise} \end{cases}$$

and if X is an assertion,

$$\mathbb{I}_X = \begin{cases} 0 & \text{if } X \\ +\infty & \text{otherwise} \end{cases}$$

Note that

$$\mathbb{I}_{g(x)=0} = \sup_{\lambda\in\mathbb{R}^{n_E}} \lambda^\top g(x)$$

and

$$\mathbb{I}_{h(x)\leq 0} = \sup_{\mu\in\mathbb{R}^{n_I}_+} \mu^\top h(x)$$

From constrained to min-sup formulation ♡

$$\begin{array}{lll} \underset{x\in\mathbb{R}^n}{\text{Min}} & f(x) & (P) \\ \text{s.t.} & g_i(x)=0 & \forall i\in[n_E] \\ & h_j(x)\le 0 & \forall j\in[n_I] \end{array}$$

Is equivalent to

$$\underset{x\in\mathbb{R}^n}{\text{Min}} \quad f(x)+\mathbb{I}_{g(x)=0}+\mathbb{I}_{h(x)\le 0}$$

or

$$\underset{x\in\mathbb{R}^n}{\text{Min}} \quad f(x)+\sup_{\lambda\in\mathbb{R}^{n_E}} \lambda^\top g(x)+\sup_{\mu\in\mathbb{R}^{n_I}_+} \mu^\top h(x)$$

which is usually written

$$\underset{x\in\mathbb{R}^n}{\text{Min}} \quad \sup_{\lambda,\mu\ge 0} \quad \underbrace{f(x)+\lambda^\top g(x)+\mu^\top h(x)}_{:=\mathcal{L}(x;\lambda,\mu)}$$

Lagrangian duality ♡

To a (primal) problem (no convexity or regularity assumptions here)

$$\begin{array}{lll} (P) \quad \underset{x\in\mathbb{R}^n}{\text{Min}} & f(x) & \\ \text{s.t.} & g_i(x)=0 & \forall i\in[n_E] \\ & h_j(x)\le 0 & \forall j\in[n_I] \end{array}$$

we associate the Lagrangian

$$\mathcal{L}(x;\lambda,\mu) := f(x)+\lambda^\top g(x)+\mu^\top h(x)$$

such that

$$(P) \qquad \underset{x\in\mathbb{R}^n}{\text{Min}} \quad \sup_{\lambda,\mu\ge 0} \quad \mathcal{L}(x;\lambda,\mu)$$

The dual problem is defined as

$$(D) \qquad \underset{\lambda,\mu\ge 0}{\text{Max}} \quad \inf_{x\in\mathbb{R}^n} \quad \mathcal{L}(x;\lambda,\mu)$$

Weak duality

By the min-max duality, we easily see that

$$\text{val}(D)\le \text{val}(P).$$

Further any admissible dual multipliers $\lambda\in\mathbb{R}^{n_E}$ $\mu\in\mathbb{R}^{n_I}_+$ yields a lower bound:

$$g(\lambda,\mu) := \inf_{x\in\mathbb{R}^n} \mathcal{L}(x;\lambda,\mu)\le \text{val}(D)\le \text{val}(P)$$

Obviously, any admissible solution $x\in\mathbb{R}^n$ (i.e. such that $g(x)=0$ and $h(x)\le 0$), yields an upper bound

$$\text{val}(P)\le f(x)=\sup_{\lambda,\mu\ge 0} \mathcal{L}(x;\lambda,\mu)$$

Min-Max duality

Recall the generic primal problem of the form

$$p^\star := \underset{x\in\mathcal{X}}{\text{Min}} \quad \sup_{y\in\mathcal{Y}} \quad \Phi(x,y)$$

with associated dual

$$d^\star := \underset{y\in\mathcal{Y}}{\text{Max}} \quad \inf_{x\in\mathcal{X}} \quad \Phi(x,y).$$

Recall that the duality gap $p^\star-d^\star\ge 0$.
We say that we have strong duality if $d^\star=p^\star$.

Saddle point

Definition

Let $\Phi : \mathcal{X} \times \mathcal{Y} \to \overline{\mathbb{R}}$ be any function. $(x^\sharp, y^\sharp)$ is a (local) saddle point of Φ on $\mathcal{X} \times \mathcal{Y}$ if

- $x^\sharp$ is a (local) minimum of $x \mapsto \Phi(x, y^\sharp)$.
- $y^\sharp$ is a (local) maximum of $y \mapsto \Phi(x^\sharp, y)$.

If there exists a global Saddle Point $(x^\sharp, y^\sharp)$ of Φ, then there is strong duality, $x^\sharp$ is an optimal primal solution and $y^\sharp$ an optimal dual solution, i.e.

$$p^\star = d^\star = \Phi(x^\sharp, y^\sharp).$$

Sufficient conditions for saddle point ♢

Theorem (Von Neumann Minimax Theorem)

If

- $\mathcal{X}$ *and* $\mathcal{Y}$ *are convex, one of them is compact*
- Φ *is continuous on* $\mathcal{X} \times \mathcal{Y}$
- $\Phi(\cdot, y)$ *is convex for all* $y \in \mathcal{Y}$
- $\Phi(x, \cdot)$ *is concave for all* $x \in \mathcal{X}$

then there exists a saddle point of Φ
$\rightsquigarrow$ *we can exchange "Min" and "Max".*

Slater's conditions for convex optimization ♡

Consider the following convex optimization problem

$$\begin{array}{lll} (P) & \underset{x \in \mathbb{R}^n}{\text{Min}} & f(x) \\ & \text{s.t.} & Ax = b \\ & & h_j(x) \leq 0 \qquad \forall j \in [n_I] \end{array}$$

We say that a point x^s such that $Ax^s = b$, $x^s \in \mathrm{ri}(\mathrm{dom}(f))$[1], and $h_j(x^s) < 0$ for all $j \in [n_I]$, is a Slater's point.

Theorem

If (P) *is convex (i.e.* f *and* h_j *are convex), and there exists a Slater's point then there is strong (Lagrangian) duality.*

Further if (P) *admits an optimal solution* $x^\sharp$ *then* $\mathcal{L}$ *admits a saddle point* $(x^\sharp, \lambda^\sharp)$*, and* $\lambda^\sharp$ *is an optimal solution to* (D).

[1] for extended function f

Perturbed problem ♢

We consider the following perturbed problem

$$\begin{array}{lll} v(p,q) = & \underset{x \in \mathbb{R}^n}{\text{Min}} & f(x) \\ & \text{s.t.} & g(x) = p \\ & & h(x) \leq q \end{array}$$

In particular we have $v(0,0) = \mathrm{val}(P)$.
By duality,

$$v(p,q) \geq d(p,q) = \sup_{\lambda, \mu \geq 0} \inf_x f(x) + \lambda^\top (g(x) - p) + \mu^\top (h(x) - q).$$

In particular, d is convex as a supremum of convex functions.

Marginal interpretation of the dual multiplier ♡

Assume that (P) is convex, and satisfies the Slater's qualification condition. In particular $v(0,0) = d(0,0)$.
Let (λ, μ) be optimal multipliers of (P).
We have, for any $x_{p,q}$ admissible for the perturbed problem, that is such that $g(x_{p,q}) = p$, $h(x_{p,q}) \leq q$,

$$\begin{aligned}\mathrm{val}(P) = v(0,0) &= \inf_x f(x) + \lambda^\top g(x) + \mu^\top h(x) \\ &\leq f(x_{p,q}) + \lambda^\top g(x_{p,q}) + \mu^\top h(x_{p,q}) \\ &\leq f(x_{p,q}) + \lambda^\top p + \mu^\top q\end{aligned}$$

In particular we have,

$$v(p,q) = \inf_{x_{p,q}} f(x_{p,q}) \geq v(0,0) - \lambda^\top p - \mu^\top q$$

which reads

$$-(\lambda, \mu) \in \partial v(0,0)$$

Exercise

♣ Exercise: Consider the following problem, for $b \in \mathbb{R}$,

$$\begin{aligned}\underset{x \in \mathbb{R}}{\text{Min}} \quad & x^2 \\ \text{s.t.} \quad & x \leq b\end{aligned}$$

1. Does there exist an optimal multiplier?
2. Without solving the dual, give the optimal multiplier μ_b.

KKT conditions ♡

Recall the first order KKT conditions for our problem (P)

$$\begin{aligned}&\nabla f(x) + A^\top \lambda + \sum_{j=1}^{n_I} \mu_j \nabla h_j(x) = 0 \\ &Ax = b, \quad h(x) \leq 0 \\ &\lambda \in \mathbb{R}^{n_E}, \quad \mu \in \mathbb{R}^{n_I}_+ \\ &\mu_j g_j(x) = 0 \qquad\qquad \forall j \in [n_I]\end{aligned}$$

Further, recall that

- the existence of a Slater's point in a convex problem ensures constraints qualifications,
- first order conditions are sufficient for convex problems.

KKT and duality ♢

If (P) is convex and there exists a Slater's point. Then the following assertions are equivalent:

1. $x^\sharp$ is an optimal solution of (P),
2. $(\exists \lambda^\sharp$ such that$)$ $(x^\sharp, \lambda^\sharp)$ is a saddle point of $\mathcal{L}$,
3. $(\exists \lambda^\sharp$ such that$)$ $(x^\sharp, \lambda^\sharp)$ satisfies the KKT conditions.

Recovering KKT conditions from Lagrangian duality

$$(P) \quad \begin{array}{ll} \underset{x\in\mathbb{R}^n}{\text{Min}} & f(x) \\ \text{s.t.} & A(x) = b \\ & h_j(x) \leq 0 \qquad \forall j \in [n_I] \end{array}$$

with associated Lagrangian

$$\mathcal{L}(x;\lambda,\mu) := f(x) + \lambda^\top (Ax - b) + \mu^\top h(x)$$

The KKT conditions can be seen as:

1. $\nabla_x \mathcal{L}(x;\lambda,\mu) = 0$ (Lagrangian minimized in x)
2. $g(x) = 0$, $h(x) \leq 0$ (x primal admissible, also obtained as $\nabla_\lambda \mathcal{L} = 0$)
3. $\mu \geq 0$ ((λ,μ) dual admissible)
4. $\mu_j = 0$ or $h_j(x) = 0$, for all $j \in [n_I]$ (complementarity constraint $\rightsquigarrow 2^{n_I}$ possibilities).

Complementarity condition and marginal value interpretation

Consider a convex problem satisfying Slater's condition.
Recall that $-\mu^\sharp \in \partial v(0)$ where $v(p)$ is the value of the perturbed problem. From this interpretation, we can recover the complementarity condition

$$\mu_j = 0 \qquad \text{or} \qquad h_j(x) = 0$$

Indeed, let x be an optimal solution.

- If constraint j is not saturated at x (i.e $h_j(x) < 0$), we can marginally move the constraint without affecting the optimal solution, and thus the optimal value. In particular, it means that $\mu_j = 0$.
- If $\mu_j \neq 0$, it means that marginally moving the constraint changes the optimal value and thus the optimal solution. In particular, constraint j must be saturated, i.e $h_j(x) = 0$.

What you have to know

- Weak duality: $\sup \inf \Phi \leq \inf \sup \Phi$
- Definition of the Lagrangian $\mathcal{L}$
- Definition of primal and dual problem

$$\underbrace{\max_{\lambda,\mu} \inf_{x} \mathcal{L}(x;\lambda,\mu)}_{\text{Dual}} \leq \underbrace{\min_{x} \sup_{\lambda,\mu} \mathcal{L}(x;\lambda,\mu)}_{\text{Primal}}$$

- Marginal interpretation of the optimal multipliers

What you really should know

- A saddle point of $\mathcal{L}$ is a primal-dual optimal pair
- Sufficient condition of strong duality under convexity (Slater's)

What you have to be able to do

- Turn a constrained optimization problem into an unconstrained Min sup problem through the Lagrangian
- Write the dual of a given problem
- Heuristically recover the KKT conditions from the Lagrangian of a problem

What you should be able to do

- Get lower bounds through duality

Optimization and algorithms

V. Leclère (ENPC)

April 3rd, 2026

Why should I bother to learn this stuff?

- Being able to recognize the type of problem is the first step toward finding the right tool to address it.
- Having an idea of the tools available to you will help choose one.
- $\Longrightarrow$ useful for any engineer (or intern) that might have to model and then solve a practical optimization problem.

Why bother with classes of optimization problems?

Consider a function $f : X \to \mathbb{R}$, and the following optimization problem

$$\begin{aligned} \min_{x \in \mathbb{R}^n} \quad & f(x) \\ \text{s.t.} \quad & x \in X \end{aligned}$$

Solving this problem can be more or less hard depending on the class in which f and $X \subset \mathbb{R}^n$[1] belongs.
Determining in which class a problem belongs is quite important:

- some problems can be solved for n of order 10 at most, other for n of order 10^6 or more;
- the methodological approach to tackle different problems vary wildly;
- the numerical tools (e.g. solvers) also...

You must be able to (roughly) classify correctly the problem you face, in order to know what can be done or not.

[1] There is also an important theory of optimization where X is not contained in a finite-dimensional space (PDE control, calculus of variations), which will not be discussed here.

Classification with respect to the objective function f ♡

- f linear is the simplest case
- f quadratic is a very important case, simple if f is convex
- f smooth (e.g. $\mathcal{C}^2$) allow to use first and second order information on f
- (f polynomial is a special case, with specific algorithms)
- f convex implies that any local minimum is a global minimum

In a convex setting, finding global optima is tractable and certifiable. Otherwise, it is generally hard. We often settle for local optima or stationary points. There exists other specific cases where global optima can be found (e.g. some non-convex quadratic problems) but they are rare.

The algorithms we present are mainly for smooth functions. Convergence theory will be done in the convex case.

Classification with respect to the constraint set ♡

- $X = \mathbb{R}^n$, is known as unconstrained optimization.
- $X = \{x \in \mathbb{R}^n \mid Ax = b\}$, can be cast, up to reparametrization, as unconstrained optimization. It might be more efficient to directly deal with the constraints.
- $X = \{x \in \mathbb{R}^n \mid \underline{x}_i \leq x_i \leq \bar{x}_i, \quad \forall i \in [n]\}$ is the box constrained optimization.
- $X = \{x \in \mathbb{R}^n \mid Ax \leq b\}$ is a polyhedron.
- X convex, generally given as $\{x \in \mathbb{R}^n \mid Ax = b, \quad h_j(x) \leq 0, \quad \forall j \in [n_I]\}$ with h_j convex.
- If X is a finite set we speak of combinatorial optimization.
- X can also be non-convex but smooth (e.g. a manifold)

A few comments:

- Unconstrained optimization is by far easier.
- Box constraints, and sometimes spherical constraints, are easy.
- Polyhedral constraints indicate LP-based methods.
- Integer constraints make the problem a lot harder and change the nature of the optimization methods.

Least-square problem (LS)

$$\underset{x \in \mathbb{R}^n}{\text{Min}} \quad \|Ax - b\|_2$$

- Equivalent *in the sense of minimizers* to $\text{Min}_{x \in \mathbb{R}^n} \|Ax - b\|_2^2$ (since $t \mapsto t^2$ increases on $\mathbb{R}_+$)
- $\rightsquigarrow$ convex, **not everywhere differentiable** (nondiff. when $Ax = b$); the squared form is convex **and smooth**
- Explicit solution via linear algebra (QR / SVD) in medium scale;
- Large-scale: iterative methods (LSQR / CG), exploiting sparsity / matrix-free products
- Typical scale (sparse, good structure): up to $n \sim 10^6$–10^8

Exercise: functional approximation

♣ Exercise: We consider a physical function Φ that is approximated as the superposition of multiple simple phenomena (e.g. waves). Each simple phenomenon $p \in [P]$ is represented by a function $\Phi_p : \mathbb{R}^d \to \mathbb{R}$.

We have data points $(x^k, y^k)_{k \in [n]}$, and want to find the Φ that match *at best* the data while being a linear combination of Φ_p.

Propose a least-square regression that answers this question.

Linear optimization problem (LP) ♡

$$\begin{aligned} \underset{x \in \mathbb{R}^n}{\text{Min}} \quad & c^\top x \\ \text{s.t.} \quad & Ax = b \\ & A'x \leq b' \end{aligned}$$

- convex problem with linear objective and polyhedral constraint set
- a rare case where exact solution can be obtained
- easily solved through dedicated code, open-source (e.g. GLPK, HiGHS) or proprietary[2] (e.g. CPLEX, Gurobi)
- can be solved for $nnz \approx 10^7$ (non-zeros coefficients of A)
- very important case in practice and as a subroutine for other problems
- two main (class of) algorithms:
  - simplex algorithm (seen in 1A)
  - interior point method (discussed later in this course)

[2] Licences are expensive(!!) but free for students!

Quadratic optimization problem (QP) ♡

$$\begin{array}{ll} \underset{x\in\mathbb{R}^n}{\text{Min}} & \frac{1}{2}x^\top Qx + c^\top x \\ \text{s.t.} & Ax = b \\ & A'x \leq b' \end{array}$$

- quadratic objective and polyhedral constraint set
- exact solution can be obtained
- easily solved if $Q \succeq 0$, hard otherwise
- can be solved for $nnz \approx 10^6$ (convex case, good structure)

Exercise: Lasso as QP

♣ Exercise: A classical extension of the least-square problem, which has strong theoretical and practical interest is the LASSO problem

$$\underset{x\in\mathbb{R}^p}{\text{Min}} \quad \|Ax - y\|^2 + \lambda\|x\|_1$$

Show that this problem can be cast as a QP problem.

Quadratically constrained quadratic problem (QCQP)

$$\begin{array}{lll} \underset{x\in\mathbb{R}^n}{\text{Min}} & \frac{1}{2}x^\top Qx + c^\top x & \\ \text{s.t.} & \frac{1}{2}x^\top P_i x + q_i^\top x \leq b_i & \forall i \in [k] \\ & Ax = b & \\ & A'x \leq b' & \end{array}$$

- Reasonably easy if convex (i.e if Q and P_i are positive semidefinite)
- can be solved for $n \approx 10^5$
- less important than previous examples

Exercise: binary optimization is equivalent to QCQP

♣ Exercise: Consider the following optimization problem.

$$\begin{array}{lll} \underset{x\in\mathbb{R}^n}{\text{Min}} & c^\top x & \\ \text{s.t.} & Ax = b & \\ & x_i \in \{0,1\} & \forall i \in I \end{array}$$

Write this problem as a QCQP. Is it convex ?

Second order cone problem (SOCP)

$$\begin{array}{ll} \underset{x\in\mathbb{R}^n}{\text{Min}} & c^\top x \\ \text{s.t.} & \|A_i x + b_i\|_2 \leq c_i^\top x + d_i \qquad \forall i \in [k] \\ & Ax = b \\ & A'x \leq b' \end{array}$$

- convex problem
- can be solved for $nnz \geq 10^6$, through most "linear" solver, relying on interior points methods
- equivalent to convex QCQP
- extend the modeling power of LP
- appears naturally in robust optimization

Exercise: robust linear programming

♠ Exercise: Consider the following robust linear program

$$\begin{array}{ll} \underset{x\in\mathbb{R}^n}{\text{Min}} & c^\top x \\ \text{s.t.} & (a_i + R_i\delta_i)^\top x \leq b_i \qquad \forall \|\delta_i\|_2 \leq 1, \quad \forall i \in [m] \end{array}$$

Write this problem as a SOCP.

Semi definite programming (SDP) ♢

$$\begin{array}{ll} \underset{X\in S_n(\mathbb{R})}{\text{Min}} & \text{tr}\left(CX\right) \\ \text{s.t.} & A(X) = b \\ & X \succeq 0 \end{array}$$

where X, and C are symmetric matrices, and $A : S_n \to \mathbb{R}^m$ a linear mapping.

- convex problem
- can be solved for $nnz \geq 10^3$, through some "linear" solver, relying on interior points methods
- contains SOCP
- limited in size in part because the number of actual variables is n^2

Unconstrained convex-differentiable optimization ♡

$$\underset{x}{\text{Min}} \quad f(x)$$

where f is convex, finite, and differentiable.

- Iterative algorithm yields ε-solution
- Solutions are global due to convexity
- Complexity theory is well understood: maximum theoretical speed, and algorithms matching this speed
- Convergence speed is better under strong convexity assumptions
- Can be solved for $n \geq 10^4$

$\rightsquigarrow$ this is where we will spend most of our time

Unconstrained convex non-differentiable optimization

$$\min_x \quad f(x)$$

where f is convex and finite.

- Iterative algorithm yields ε-solution
- Solutions are global due to convexity
- Complexity theory is well understood: maximum theoretical speed, and algorithms matching this speed
- Can be solved for $n \geq 10^3$

Unconstrained differentiable optimization

$$\min_x \quad f(x)$$

where f is differentiable.

- Iterative algorithm yields ε stationary point (i.e $\|\nabla f(x)\| \leq \varepsilon$)
- Algorithms are mostly the same as in convex differentiable setting, but the theory is more involved
- Can find a stationary point for $n \geq 10^4$

$\rightsquigarrow$ most algorithms presented in this course can be used to hopefully get to a locally optimal point.

Constrained convex optimization

$$\begin{aligned} \min_x \quad & f(x) \\ \text{s.t.} \quad & x \in X \end{aligned}$$

where X is a convex set.

- Easiest if X is a box or ball
- Specific approach relying on LP if X is a polyhedron
- Various methods in the generic case:
  - projection
  - feasible direction
  - constraint penalization
  - dualization

Combinatorial problem

$$\min_{x \in X} \quad f(x)$$

where X is a finite set.

- This roughly represent the problem of combinatorial optimization.
- X being finite you can, in theory, test all possibilities and choose the best. However, this brute force approach is often unpractical due to the size of X.
- Even if an exact solution can be obtained, it is not often the case.
- Finding lower-bounds is interesting to understand how far your current solution is from the optimum.
- Practical methods are often *matheuristics* or *meta-heuristics* adapted to the specificity of the problem.
- Problems are often very hard, and practical solvability depends on the specific problem structure.

Mixed Integer Linear Programming ♡

$$\begin{aligned} \min_{x\in\mathbb{R}^n} \quad & c^\top x \\ \text{s.t.} \quad & Ax = b, x \geq 0 \\ & x_i \in \mathbb{N} \qquad\qquad \forall i \in I \subset [n] \end{aligned}$$

- A very important class of problem, with huge modeling power.
- By order of difficulty we distinguish: continuous variables, binary variables and integer variables.
- Exact solution methods rely on the idea of branch and cut. (`https://www.youtube.com/watch?v=2zKCQ03JzOY` (13'))
- Very powerful (commercial) solvers (like Gurobi, Cplex, Mosek...) are developed and improved every year to tackle these problems. They use a mix of insightful mathematical ideas and heuristic knowledge.
- Efficiency of the solver depends on the type of problem, and the formulation of the problem.

Mixed Integer Conic Programming ♢

$$\begin{aligned} \min_{x\in\mathbb{R}^n} \quad & c^\top x \\ \text{s.t.} \quad & Ax = b \\ & x \geq 0 \\ & x \in C \\ & x_i \in \mathbb{N} \qquad\qquad \forall i \in I \subset [n] \end{aligned}$$

where C is a convex cone.

- Harder than MILP
- More recent development, thus the theory and heuristic experience is less advanced than MILP
- Numerical efficiency is quickly improving

Exercise: stock optimization

♠ Exercise: Consider that you sell a given product over T days. The demand for each day is d_t. Having a quantity x_t of items in stock have a cost (per day) of cx_t. You can order, each day, a quantity q_t, and have to satisfy the demand.

For each of the following variations: model the problem, give the class to which it belongs, and give the optimal solution if easily found.

1. Without any further constraints / specifications.
2. There is an "ordering cost": each time you order, you have to pay a fixed cost κ.
3. Instead of an "ordering cost" there is a maximum number of days at which you can order a replenishment.

Solver families and typical scale rule of thumb ♢

- **Dense second–order methods** (Newton, generic interior–point)
  - Each iteration requires factorizing an $n \times n$ matrix
  - Typical scale: $n \lesssim 10^3$ (dense), up to 10^4 with strong sparsity
  - High accuracy, few iterations
- **Sparse direct methods** (sparse Cholesky / LU inside IPM or SQP)
  - Cost dominated by fill–in and sparsity pattern
  - Typical scale: $n \sim 10^4$–10^6 in favorable cases
  - Common for LP, convex QP, SOCP
- **First–order methods** (gradient, proximal, ADMM, coordinate descent)
  - Each iteration uses gradients or simple proximal operators
  - Typical scale: $n \sim 10^5$–10^8 if operations are cheap/sparse
  - Lower accuracy, many iterations
- **Stochastic / streaming methods** (SGD and variants)
  - Use only partial data at each iteration
  - Typical scale: $n \gtrsim 10^6$, data size 10^7+ and beyond
  - Dominant in machine learning

"Ad Hoc" solution

A heuristic is an admissible, not necessarily optimal, solution to a given optimization problem. It gives upper-bounds.

- In a lot of applications, experience or good sense, can give reasonably good heuristics.
- Sometimes these heuristics can have a few parameters that can be tuned by trial and error.

♣ Exercise: In the stock optimization example, with fixed ordering cost, propose a simple heuristic.
♣ Exercise: Now assume that, in this same example, there is some uncertainty on the demand, adapt your heuristic to offer a *robustness* parameter.

Random search

A good way of obtaining *good* solutions is to randomly test multiple admissible solutions, and keep the best one.
Examples:

- exhaustive search (combinatorial)
- genetic algorithms
- simulated annealing
- particle swarm

Use case :

- hard problems (combinatorial or continuous) where finding an admissible solution is easy
- when you just want an admissible solution, if possible better than what you had

Descent methods ♡

Consider the unconstrained optimization problem

$$\min_{x \in \mathbb{R}^n} \quad f(x). \tag{1}$$

A *descent direction algorithm* is an algorithm that constructs a sequence of points $(x^{(k)})_{k \in \mathbb{N}}$, that are recursively defined with:

$$x^{(k+1)} = x^{(k)} + t^{(k)} d^{(k)} \tag{2}$$

where

- $x^{(0)}$ is the initial point,
- $d^{(k)} \in \mathbb{R}^n$ is the descent direction,
- $t^{(k)}$ is the step length.

$\rightsquigarrow$ most of this is discussed in next classes.

Descent direction

For a differentiable objective function f, $d^{(k)}$ will be a descent direction iff $\nabla f(x^{(k)}) \cdot d^{(k)} < 0$, which can be seen from a first order development:

$$f(x^{(k)} + t d^{(k)}) = f(x^{(k)}) + t \left\langle \nabla f(x^{(k)}) \, , d^{(k)} \right\rangle + o(t).$$

The most classical descent direction is $d^{(k)} = -\nabla f(x^{(k)})$, which corresponds to the gradient algorithm.

Step-size choice

The step-size $t^{(k)}$ can be:

- fixed $t^{(k)} = t^{(0)}$, for all iteration,
- optimal $t^{(k)} \in \arg\min_{t \geq 0} f(x^{(k)} + t d^{(k)})$,
- a "good" step, following some rules (e.g Armijo's rules).

Finding the optimal step size is a special case of unidimensional optimization (or line search).

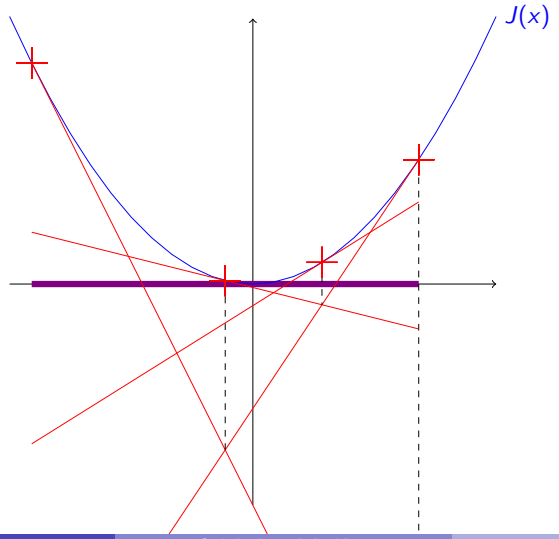
Model based methods ♡

Another class of algorithm consists in constructing a simple model of the objective function f that is optimized and then refined.
Generally speaking, model-based algorithm goes as follows:

1. Solve $\min_{x \in X} f^k(x)$
2. Update model f^k into f^{k+1}

This approach might work if

- The model problem $\min_{x \in X} f^k(x)$ is *simple*
- The model f^k locally looks like the true function f around the optimum



Kelley algorithm

Data: Convex objective function f, Compact set X, Initial point $x_0 \in X$
Result: Admissible solution $x^{(k)}$, lower-bound $\underline{v}^{(k)}$
Set $f^{(0)} \equiv -\infty$;
for $k \in \mathbb{N}$ **do**
  Compute a subgradient $g^{(k)} \in \partial f(x^{(k)})$;
  Define a cut $\mathcal{C}^{(k)} : x \mapsto f(x^{(k)}) + \langle g^{(k)}, x - x^{(k)} \rangle$;
  Update the lower approximation $f^{(k+1)} = \max\{f^{(k)}, \mathcal{C}^{(k)}\}$;
  Solve $(P^{(k)})$: $\min_{x \in X} f^{(k+1)}(x)$;
  Set $\underline{v}^{(k)} = val(P^{(k)})$;
  Select $x^{(k+1)} \in sol(P^{(k)})$;
end

Algorithm 1: Kelley's cutting plane algorithm

Trust region method

Consider an unconstrained, non-linear, smooth problem

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

The idea of trust region is based on the following two facts:

- f locally looks like it's second order limited development

$$f(x+h) = f(x) + \langle \nabla f(x), h \rangle + \frac{1}{2} h^\top \nabla^2 f(x) h + o(\|h\|^2)$$

- we know how to compute the minimum of a quadratic function on a ball.

Trust region method

The trust region method goes as follows, given a current point x^k and *trust radius* Δ^k

1. compute $f^k(x^k + h) = f(x^k) + \langle \nabla f(x^k), h \rangle + \frac{1}{2} h^\top \nabla^2 f(x^k) h$
2. solve $\min_{y \in B(x^k, \Delta^k)} f^k(y)$, with optimal solution y^k
3. compute $f(y^k)$
4. compute the *concordance* r^k as the ratio actual decrease / model decrease

$$r^k = \frac{f(x^k) - f(y^k)}{f^k(x^k) - f^k(y^k)}$$

   - If r^k is small, the model is bad and you decrease Δ^k and restart the iteration
   - If r^k is large (close to 1) update the current point $x^{k+1} = y^k$.
5. If r^k is close to one and y^k is on the boundary, increase Δ^k.

$\rightsquigarrow$ there are full books on trust region methods.

What you have to know

- Important elements defining an optimization problem : continuous/discrete, smooth/non-differentiable, convex/non-convex, linear/non-linear, constrained/unconstrained.
- Main optimization classes: LP, MILP, differentiable unconstrained, combinatorial.
- The difference between heuristic and exact methods
- Main classes of exact method : descent direction, approximation method.

What you really should know

- Other important classes of optimization problem (LS, QP, SOCP, SDP)
- Some ideas of heuristic methods (simulated annealing, genetic algorithms)
- Kelley's cutting plane algorithm
- Principle of trust region method

What you have to be able to do

- Recognise a LP / MILP
- Recognise a (convex) differentiable optimization problem, constrained or not

What you should be able to do

- know how to use a "lift" variable, e.g.

$$\min_x \quad \max(f_1(x), f_2(x)) = \quad \min_{x,z} \quad z$$
$$\text{s.t.} \quad f_1(x) \leq z$$
$$f_2(x) \leq z$$

Gradient algorithms

V. Leclère (ENPC)

April 17th, 2026

Why should I bother to learn this stuff?

- Gradient algorithms are among the easiest, most robust optimization algorithms. They are not numerically efficient, but numerous more advanced algorithms are built on them.
- Conjugate gradient algorithm(s) are efficient methods for (quasi)-quadratic function. They are in particular used for approximately solving large linear systems.
- $\Longrightarrow$ useful for comprehension of
  - more advanced continuous optimization algorithms
  - machine learning training methods
  - numerical methods for solving discretized PDE

A word on solution

- In this lecture, we are going to address unconstrained, finite dimensional, non-linear, smooth, optimization problem.
- In continuous non-linear (and non-quadratic) optimization, we cannot expect to obtain an *exact* solution. We are thus looking for approximate solutions.
- By solution, we generally mean local minimum.[1]
- The speed of convergence of an algorithm is thus determining an upper bound on the number of iterations required to get an ε-solution, for $\varepsilon > 0$:

$$f(x) \leq v^\sharp + \varepsilon$$

where $v^\sharp$ is the optimal value of the problem.

[1] Sometimes just stationary points. Equivalent to global minimum in the convex setting.

Black-box optimization ♡

We consider the following unconstrained optimization problem

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

- The black-box model consists in considering that we only know the function f through an oracle, that is a way of computing information on f at a given point x.
- Oracle gives local information on f. Oracles are generally given as user-defined code.
  - A *zeroth* order oracle only returns the value $f(x)$.
  - A *first* order oracle returns both $f(x)$ and $\nabla f(x)$.
  - A *second* order oracle returns $f(x)$, $\nabla f(x)$ and $\nabla^2 f(x)$.
- By opposition, structured optimization leverage more knowledge on the objective function f. Classical models are
  - $f(x) = \sum_{i=1}^N f_i(x)$;
  - $f(x) = f_0(x) + \lambda g(x)$, where $f_0(x)$ is smooth and g is "simple", typically $g(x) = \|x\|_1$;
  - ...

Descent methods

Consider the unconstrained optimization problem

$$v^\sharp = \min_{x\in\mathbb{R}^n} \quad f(x).$$

A *descent direction algorithm* is an algorithm that constructs a sequence of points $(x^{(k)})_{k\in\mathbb{N}}$, that are recursively defined with:

$$x^{(k+1)} = x^{(k)} + t^{(k)} d^{(k)}$$

where

- $x^{(0)}$ is the initial point,
- $d^{(k)} \in \mathbb{R}^n$ is the descent direction,
- $t^{(k)}$ is the step length.

For most of the analysis, we will assume f to be (strongly) convex, but the algorithms presented are often used in a non-convex setting.

To complete the algorithm, we need a stopping test, generally testing that $\|\nabla f(x^{(k)})\|$ is small enough.

Descent direction algorithms ♡

For a differentiable objective function f, $d^{(k)}$ will be a descent direction iff $\nabla f(x^{(k)}) \cdot d^{(k)} < 0$, which can be seen from a first order development:

$$f(x^{(k)} + t d^{(k)}) = f(x^{(k)}) + t\left\langle \nabla f(x^{(k)}), d^{(k)} \right\rangle + o(t).$$

The most classical descent direction are[2]

1. $d^{(k)} = -\nabla f(x^{(k)})$ (gradient)
2. $d^{(k)} = -\nabla f(x^{(k)}) + \beta^{(k)} d^{(k-1)}$ (conjugate gradient)
3. $d^{(k)} = -\left[\nabla^2 f(x^{(k)})\right]^{-1} \nabla f(x^{(k)})$ (Newton)
4. $d^{(k)} = -W^{(k)} \nabla f(x^{(k)})$ (Quasi-Newton)
   where $W^{(k)} \approx \left[\nabla^2 f(x^{(k)})\right]^{-1}$.

[2] they will be discussed at length during the course.

Step-size choice ♡

The step-size $t^{(k)}$ can be:

- fixed $t^{(k)} = t^{(0)}$,
  - too small and it will take forever
  - too large and it won't converge
- optimal $t^{(k)} \in \arg\min_{\tau\geq 0} f(x^{(k)} + \tau d^{(k)})$,
  - computing it requires solving an unidimensional problem
  - might not be worth the computation
- a backtracking or receeding step choice[3], for given $\tau^0 > 0, \alpha \in]0, 0.5[, \beta \in]0, 1[$,
  1. $\tau = \tau^0$
  2. if $f(x^{(k)} + \tau d^{(k)}) \leq f(x^{(k)}) + \alpha\tau \nabla f(x^{(k)})^\top d^{(k)}$: $t^{(k)} = \tau$, STOP
  3. $\tau \leftarrow \beta\tau$, go back to 2.
  - start with an "optimist" step τ_0
  - automatically adapts to ensure convergence
  - more complex procedure exists

[3] There exists a lot of other alternatives

Strong convexity definition(s) ♡

Recall that $f : \mathbb{R}^n \to \mathbb{R}$ is m-strongly convex[4] iff

$$f(tx + (1-t)y) \leq tf(x) + (1-t)f(y) - \frac{m}{2} t(1-t)\|y - x\|^2, \quad \forall x, y, \quad \forall t \in]0, 1[$$

If f is differentiable, it is m-strongly convex iff

$$f(y) \geq f(x) + \left\langle \nabla f(x), y - x \right\rangle + \frac{m}{2}\|y - x\|^2, \qquad \forall y, x$$

If f is twice differentiable, it is m-strongly convex iff

$$mI \preceq \nabla^2 f(x) \qquad \forall x$$

iff

$$m \leq \lambda \qquad \forall \lambda \in sp(\nabla^2 f(x)), \quad \forall x$$

$\rightsquigarrow$ this last characterization is the most useful for our analysis.

[4] A strongly convex function is a m-strongly convex function for some $m > 0$

Bounding the Hessian

Consider a m-strongly convex $\mathcal{C}^2$ function, and $x^{(0)} \in \operatorname{dom} f$. Denote $S := \operatorname{lev}_{f(x_0)}(f) = \{x \in \mathbb{R}^n \mid f(x) \leq f(x_0)\}$.

As f is a strongly convex function S is bounded.

As $\nabla^2 f$ is continuous, there exists $M > 0$ such that, $\|\nabla^2 f(x)\| \leq M$, for all $x \in S$.

Thus we have, for all $x \in S$,

$$mI \preceq \nabla^2 f(x) \preceq MI$$

Or equivalently

$$m \leq \lambda_{min}(\nabla^2 f(x)) \leq \lambda_{\max}(\nabla^2 f(x)) \leq M \qquad \forall x \in S$$

Strongly convex suboptimality certificate

Let f be a m-strongly convex $\mathcal{C}^2$ function. We have

$$f(y) \geq f(x) + \langle \nabla f(x), y - x \rangle + \frac{m}{2}\|y - x\|^2, \qquad \forall y, x$$

The under approximation is minimized, for a given x, for $y^\sharp = x - \frac{1}{m}\nabla f(x)$, yielding

$$f(y) \geq f(x) - \frac{1}{2m}\|\nabla f(x)\|^2 \qquad \forall y$$

$$v^\sharp + \frac{1}{2m}\|\nabla f(x)\|^2 \geq f(x) \qquad \forall x$$

Thus we obtain the following sub-optimality certificate

$$\|\nabla f(x)\| \leq \sqrt{2m\varepsilon} \quad \Longrightarrow \quad f(x) \leq v^\sharp + \varepsilon$$

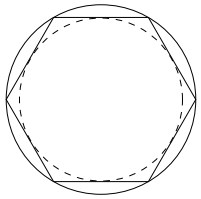
Condition numbers

For any $A \in S_n^{++}$ positive definite matrix, we define its condition number $\kappa(A) = \lambda_{max}/\lambda_{min} \geq 1$ the ratio between its largest and smallest eigenvalue.

Consider a bounded convex set C. Let D_{out} be the diameter of the smallest ball B_{out} containing C, and D_{in} be the diameter of the largest ball B_{in} contained in C.

Then the condition number of C is

$$\operatorname{cond}(C) = \left(\frac{D_{out}}{D_{in}}\right)^2$$



Condition number of sublevel set

We have, for all $x \in S$,

$$mI \preceq \nabla^2 f(x) \preceq MI$$

thus

$$\kappa(\nabla^2 f(x)) \leq M/m$$

Further,

$$v^\sharp + \frac{m}{2}\|x - x^\sharp\|^2 \leq f(x) \leq v^\sharp + \frac{M}{2}\|x - x^\sharp\|^2$$

For any $v^\sharp \leq \alpha \leq f(x_0)$, we have

$$B\left(x^\sharp, \sqrt{2(\alpha - v^\sharp)/M}\right) \subset \operatorname*{lev}_\alpha f \subset B\left(x^\sharp, \sqrt{2(\alpha - v^\sharp)/m}\right)$$

and thus

$$\operatorname{cond}(C_\alpha) \leq M/m$$

Gradient descent ♡

- The gradient descent algorithm is a first-order descent direction algorithm with $d^{(k)} = -\nabla f(x^{(k)})$.
- That is, with an initial point x_0, we have

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

- The three step-size choices (fixed, optimal and receeding) lead to variations of the algorithm.

- This algorithm is slow, but robust in the sense that it often ends up converging.
- Most implementations of advanced algorithms have fail-safe procedures that default to a gradient step when something goes wrong for numerical reasons.
- It is the basis of the stochastic-gradient algorithm, which is used (in advanced form) to train ML models.

Steepest descent algorithm ♢

- Using the linear approximation $f(x^{(k)} + h) = f(x^{(k)}) + \nabla f(x^{(k)})^\top h + o(\|h\|_✠)$, it is quite natural to look for the steepest descent direction, that is

$$d^{(k)} \in \arg\min_h \quad \left\{ \nabla f(x^{(k)})^\top h \quad | \quad \|h\|_✠ \leq 1 \right\}$$

- Here $\|\cdot\|_✠$ could be any norm on $\mathbb{R}^n$.
  - If $\|\cdot\|_✠ = \|\cdot\|_2$, the steepest descent is a gradient step, i.e. proportional to $-\nabla f(x^{(k)})$.
  - If $\|\cdot\|_✠ = \|\cdot\|_P$, $\|x\|_✠ = \|P^{1/2}x\|_2$ for some $P \in S^n_{++}$, then the steepest descent is $-P^{-1}\nabla f(x^{(k)})$. In other words, a steepest descent step is a gradient step done on a problem after a change of variable $\bar{x} = P^{1/2}x$.
  - If $\|\cdot\|_✠ = \|\cdot\|_1$, then the steepest descent can be chosen along a single coordinate, leading to the coordinate descent algorithm.

♠ Exercise: Prove these results.

Convergence results - convex case ♢

Assume that f is such that $0 \preceq \nabla^2 f \preceq MI$.

Theorem

The gradient algorithm with fixed step size $t^{(k)} = \frac{1}{M}$ satisfies

$$f(x^{(k)}) - v^\sharp \leq 2\frac{M\|x^{(0)} - x^\sharp\|^2}{k-1} = O(1/k)$$

$\rightsquigarrow$ this is a *sublinear* rate of convergence.

Convergence results - strongly convex case ♢

Assume that f is such that $mI \preceq \nabla^2 f \preceq MI$, with $m > 0$. Define the conditioning factor $\kappa = M/m$.

Theorem

If $x^{(k)}$ is obtained from the optimal step, we have

$$f(x^{(k)}) - v^\sharp \leq C^k (f(x_0) - v^\sharp), \qquad C = 1 - 1/\kappa$$

If $x^{(k)}$ is obtained by receeding step size we have

$$f(x^{(k)}) - v^\sharp \leq C^k (f(x_0) - v^\sharp), \qquad C = 1 - \min\left\{2m\alpha, 2\beta\alpha\right\}/\kappa$$

$\rightsquigarrow$ linear rate of convergence.

Solving a linear system I

The conjugate gradient algorithm stems from looking for numerical solutions to the linear equation

$$Ax = b$$

- Never, ever, compute A^{-1} to solve a linear system.
- Classical algebraic method do a methodological factorization of A to obtain the (exact) value of x.
- These methods are in $O(n^3)$ operations. They only yield a solution[5] at the end of the algorithm.
- This qualify as dense method, for sparse matrices filling tends to occur.
- Iterative methods (like conjugate gradient) build a sequence of approximate solutions $(x^{(k)})_{k\in\mathbb{N}}$.
- Each iteration is in $O(n^2)$ operations (for dense A).

[5] Exact if there was no rounding error

Solving a linear system II

Alternatively, we can look to solve

$$\min_{x\in\mathbb{R}^n} \quad f(x) := \frac{1}{2}x^\top Ax - b^\top x$$

which is a smooth, unconstrained, convex optimization problem, whose optimal solution is given by $Ax = b$.

We will assume that $A \in S^n_{++}$.
If A is non symmetric, but invertible, we could consider $A^\top Ax = A^\top b$. However, this is not numerically stable, and should be avoided in practice.[6]

[6] Use GMRES or BiCGStab instead.

Conjugate directions ♢

We say that $u, v \in \mathbb{R}^n$ are A-conjugate if they are orthogonal for the scalar product associated to A, i.e.

$$\langle u\,,\,v\rangle_A := u^\top Av = 0$$

Let $(\tilde{d}_i)_{i\in[k]}$ be a linearly independent family of vectors. We can construct a family of conjugate directions $(d_i)_{i\in[k]}$ through the Gram-Schmidt procedure (without normalization), i.e., $\tilde{d}_1 = d_1$, and

$$d_\kappa = \tilde{d}_\kappa - \sum_{i=1}^{\kappa-1} \beta_{i,\kappa} d_i$$

where

$$\beta_{i,\kappa} = \frac{\langle \tilde{d}_\kappa\,,\,d_i\rangle_A}{\langle d_i\,,\,d_i\rangle_A} = \frac{\tilde{d}_\kappa^\top Ad_i}{d_i^\top Ad_i}$$

Note: this is conceptual, not a numerically stable procedure.

Conjugate direction method for quadratic function I ♢

Consider, for $A \in S^n_{++}$

$$f(x) := \frac{1}{2}x^\top Ax - b^\top x$$

A conjugate direction algorithm is a descent direction algorithm such that,

$$x^{(k+1)} = \operatorname*{arg\,min}_{x\in x^{(1)}+E^{(k)}} \quad f(x)$$

where

$$E^{(k)} = vect(d^{(1)}, \dots, d^{(k)})$$

♠ Exercise: Denote $g^{(k)} = \nabla f(x^{(k)})$. Show that

1. ${g^{(k)}}^\top d_i = 0$ for $i < k$
2. $g^{(k+1)} = g^{(k)} + t^{(k)}Ad^{(k)}$
3. ${g^{(k)}}^\top d^{(i)} + t^{(k)}{d^{(k)}}^\top Ad^{(i)} = 0$ for $i \le k$
4. Either
   - ${g^{(k)}}^\top d^{(k)} = 0$ and $t^{(k)} = 0$
   - or ${g^{(k)}}^\top d^{(k)} < 0$ and $t^{(k)} = -\frac{{g^{(k)}}^\top d^{(k)}}{{d^{(k)}}^\top Ad^{(k)}}$

Conjugate direction method for quadratic function II ♢

Data: Linearly independent direction $\tilde{d}^{(1)}, \dots, \tilde{d}^{(n)}$, initial point $x^{(1)}$
Matrix A and vector b
for $k \in [n]$ **do**
  $d^{(k)} = \tilde{d}^{(k)} - \sum_{i=1}^{k-1} \frac{\left\langle \tilde{d}^{(k)}, d^{(i)} \right\rangle_A}{\left\langle d^{(i)}, d^{(i)} \right\rangle_A} d^{(i)}$; // A-orthogonalisation
  $t^{(k)} = -\frac{\nabla f(x^{(k)})^\top d^{(k)}}{\left\langle d^{(k)}, d^{(k)} \right\rangle_A}$; // optimal step
  $x^{(k+1)} = x^{(k)} + t^{(k)} d^{(k)}$

Algorithm 1: Conjugate direction algorithm

This algorithm is such that (for a quadratic function f)

$$x^{(k+1)} = \underset{x \in x_1 + E^{(k)}}{\arg\min} \quad f(x)$$

where

$$E^{(k)} = vect(d^{(1)}, \dots, d^{(k)})$$

Conjugate gradient algorithm - quadratic function I ♢

The conjugate gradient algorithm set $\tilde{d}^{(k)} = -\underbrace{\nabla f(x^{(k)})}_{:=g^{(k)}}$.

In particular, we obtain that $E^{(k)} = vect(g^{(1)}, \dots, g^{(k)})$, and thus

$$g^{(k)\top} g^{(i)} = 0 \qquad \forall i \neq k$$

Note that

$$g^{(i+1)} - g^{(i)} = t^{(i)} A d^{(i)}, \quad \text{thus} \quad \frac{\left\langle \tilde{d}^{(k)}, d^{(i)} \right\rangle_A}{\left\langle d^{(i)}, d^{(i)} \right\rangle_A} = \frac{(\tilde{d}^{(k)})^\top (g^{(i+1)} - g^{(i)})}{d^{(i)\top} (g^{(i+1)} - g^{(i)})}$$

Thus, through orthogonality we have

$$\begin{aligned} d^{(k)} &= \tilde{d}^{(k)} - \sum_{i=1}^{k-1} \frac{-g^{(k)\top} (g^{(i+1)} - g^{(i)})}{d^{(i)\top} (g^{(i+1)} - g^{(i)})} d^{(i)} \\ &= -g^{(k)} + \frac{g^{(k)\top} (g^{(k)} - g^{(k-1)})}{d^{(k-1)\top} (g^{(k)} - g^{(k-1)})} d^{(k-1)} = -g^{(k)} + \frac{\|g^{(k)}\|^2}{\|g^{(k-1)}\|^2} d^{(k-1)} \end{aligned}$$

Conjugate gradient algorithm - quadratic function II ♢

Data: Initial point $x^{(1)}$, matrix $A \in S_n^{++}$ and vector b
$g^{(1)} = Ax^{(1)} - b$;
$d^{(1)} = -g^{(1)}$;
for $k = 1..n$ **do**
  if $\|g^{(k)}\|_2^2$ *is small* **then**
    STOP
  ;
  $t^{(k)} = \frac{\|g^{(k)}\|_2^2}{d^{(k)\top} A d^{(k)}}$; // optimal step
  $x^{(k+1)} = x^{(k)} + t^{(k)} d^{(k)}$;
  $g^{(k+1)} = g^{(k)} + t^{(k)} A d^{(k)}$; // update gradient
  $d^{(k+1)} = -g^{(k+1)} + \frac{\|g^{(k+1)}\|_2^2}{\|g^{(k)}\|_2^2} d^{(k)}$;

Algorithm 2: Conjugate gradient algorithm - quadratic function

Conjugate gradient properties ♢

We can show the following properties, for a quadratic function,

- The algorithm finds an optimal solution in at most n iterations
- If $\kappa = \lambda_{max}/\lambda_{min}$, we have

$$\|x^{(k+1)} - x^\sharp\|_A \leq 2\left(\frac{\sqrt{\kappa} - 1}{\sqrt{\kappa} + 1}\right)^k \|x^{(1)} - x^\sharp\|_A$$

- By comparison, gradient descent with optimal step yields

$$\|x^{(k+1)} - x^\sharp\|_A \leq 2\left(\frac{\kappa - 1}{\kappa + 1}\right)^k \|x^{(1)} - x^\sharp\|_A$$

Non-linear conjugate gradient ♢

Data: Initial point $x^{(1)}$, first order oracle
$d^{(0)} = 0$;
for $k \in [n]$ **do**
  $g^{(k)} = \nabla f(x^{(k)})$;
  If $\|g^{(k)}\|_2^2$ is small : STOP;
  $d^{(k)} = -g^{(k)} + \beta^{(k)} d^{(k-1)}$;
  $t^{(k)}$ obtained by backtracking line search;
  $x^{(k+1)} = x^{(k)} + t^{(k)} d^{(k)}$;

Algorithm 3: Conjugate gradient algorithm - non-linear function

Two natural choices for β, equivalent for quadratic functions

- $\beta^{(k)} = \dfrac{\|g^{(k)}\|_2^2}{\|g^{(k-1)}\|_2^2}$ (Fletcher-Reeves)
- $\beta^{(k)} = \left(\dfrac{{g^{(k)}}^\top (g^{(k)} - g^{(k-1)})}{\|g^{(k-1)}\|_2^2} \right)^+$ (Polak-Ribière)

What you have to know

- What is a descent direction method.
- That there is a step-size choice to make.
- That there exists multiple descent direction.
- Gradient method is the slowest method, and in most case you should used more advanced method through adapted library.
- Conditionning of the problem is important for convergence speed.

What you really should know

- A problem can be pre-conditionned through change of variable to get faster results.
- Solving linear system can be done exactly through algebraic method, or approximately (or exactly) through minimization method.
- Conjugate gradient method are efficient tools for (approximately) solving a linear equation.
- Conjugate gradient works by exactly minimizing the quadratic function on an affine subspace.

What you have to be able to do

- Implement a gradient method with receeding step-size.

What you should be able to do

- Implement a conjugate gradient method.
- Use the strongly convex and/or Lipschitz gradient assumptions to derive bounds.

Newton and Quasi-Newton algorithms

V. Leclère (ENPC)

April 24th, 2026

Why should I bother to learn this stuff?

- Newton algorithm is, in theory, the best black-box algorithm for smooth strongly convex function. It is used in practice as well as a stepping step for more advanced algorithm.
- Quasi-Newton algorithms (in particular L-BFGS) are the de facto by default algorithm for most smooth black-box optimization library. Used in large scale application (e.g. weather forecast) for decades.
- $\Longrightarrow$ useful for
  - understanding the optimization software you might use as an engineer
  - understanding more advanced methods (e.g. interior points methods)
  - getting an idea of why the convergence might behave strangely in practice

Oriented sum-up of previous courses

- There are two large classes of unconstrained, exact, black-box, optimization algorithms:
  - descent direction algorithm: $x^{(k+1)} = x^{(k)} + t^{(k)} d^{(k)}$;
  - model based approach: $x^{(k+1)} = \arg\min_x f^{(k)}(x)$.
- We saw that defining a descent direction algorithm requires:
  - a direction $d^{(k)}$;
  - a step $t^{(k)}$;
  - a stopping test (e.g. $\|\nabla f(x^{(k)})\|_2 \ll 1$)
- We discussed gradient and conjugate gradient algorithms defined by $d^{(k)} = -\nabla f(x^{(k)}) + \beta^{(k)} d^{(k-1)}$:
  - convergence speed is sensitive to conditionning of the problem (i.e. if level sets are almost spherical);
  - you can precondition the problem through a change of coordinates;
  - can be interpreted as steepest descent method: $d^{(k)} = \arg\min_{\|d\|_P \leq 1} \nabla f(x^{(k)})^\top d$

Newton algorithm ♡

Let f be $\mathcal{C}^2$ such that $\nabla^2 f(x) \succ 0$[1] for all x (so in particular strictly convex).
The Newton algorithm is a descent direction algorithm with:

- $d^{(k)} = -[\nabla^2 f(x^{(k)})]^{-1} \nabla f(x^{(k)})$
- $t^{(k)} = 1$

Note that

$$\nabla f(x^{(k)})^\top d^{(k)} = -\nabla f(x^{(k)})^\top [\nabla^2 f(x^{(k)})]^{-1} \nabla f(x^{(k)}) < 0$$

(unless $\nabla f(x^{(k)}) = 0$)
$\rightsquigarrow$ $d^{(k)}$ is a descent direction.

We are now going to give multiple justifications for this direction choice.

[1] This will be relaxed later.

Second-order approximation minimization

We have

$$f(x^{(k)}+d) = f(x^{(k)}) + \nabla f(x^{(k)})^\top d + \frac{1}{2} d^\top \nabla^2 f(x^{(k)}) d + o(\|d\|^2)$$

The Newton method chooses the direction d (with step 1) that minimizes this second-order approximation, which is given by

$$\nabla f(x^{(k)}) + \nabla^2 f(x^{(k)}) d^{(k)} = 0$$

$\rightsquigarrow$ The Newton method can be seen as a model-based method, where the model at iteration k is simply the second-order approximation.

$\rightsquigarrow$ A trust region method with confidence radius $+\infty$ is simply the Newton method.

Alternative views

Steepest descent with adaptive norms

- The Newton direction $d^{(k)}$ is the steepest descent direction for the quadratic norm associated to $\nabla^2 f(x^{(k)})$:

$$d^{(k)} = \arg\min_d \left\{ \nabla f(x^{(k)})^\top d \quad | \quad \|d\|_{\nabla^2 f(x^{(k)})} \leq 1 \right\}$$

- Recall that the steepest gradient descent for a quadratic norm $\|\cdot\|_P$ converges rapidly if the condition number of the Hessian, after a change of coordinate, is small.
- In particular a good choice near $x^\sharp$ is $P = \nabla^2 f(x^\sharp)$.

$\rightsquigarrow$ fast around $x^\sharp$

Solution of linearized optimality conditions

The optimality condition is given by

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

We can linearize it as

$$\nabla f(x^{(k)}+d) \approx \nabla f(x^{(k)}) + \nabla^2 f(x^{(k)}) d = 0$$

And the Newton step $d^{(k)}$ is the solution of this linearization.

Affine invariance

- Recall that gradient and conjugate gradient methods can be accelerated through smart affine changes of variables (pre-conditioning).
- It is not the same for the Newton method:
  - Let A be an invertible matrix, and denote $y = Ax + b$, and $\tilde{f} : x \mapsto f(Ax+b)$.
  - $\nabla \tilde{f}(y) = A \nabla f(x)$ and $\nabla^2 \tilde{f}(y) = A^\top \nabla^2 f(x) A$
  - The Newton step for $\tilde{f}$ is thus

$$d_y = -(A^\top \nabla^2 f(x) A)^{-1} A \nabla f(x) = -A^{-1} (\nabla^2 f(x))^{-1} \nabla f(x) = A^{-1} d_x$$

  - Consequently

$$x^{(k+1)} - x^{(k)} = A(y^{(k+1)} - y^{(k)})$$

➥ The Newton method does not really benefit from preconditioning!

Damped Newton algorithm

Data: Initial point $x^{(0)}$, second-order oracle, error $\varepsilon > 0$.
while $\|\nabla f(x^{(k)})\| \geq \varepsilon$ **do**
  Solve for $d^{(k)}$

$$\nabla^2 f(x^{(k)}) d^{(k)} = -\nabla f(x^{(k)})$$

  Compute $t^{(k)}$ by backtracking line-search, starting from $t = 1$;
  $x^{(k+1)} = x^{(k)} + t^{(k)} d^{(k)}$

Algorithm 1: Damped Newton algorithm

- The Newton algorithm with fixed step size $t = 1$ may fail far from the optimum, and you should always use a backtracking line-search.
- If the function is not strictly convex the Newton direction is not necessarily a descent direction, and you should check for it (and default to a gradient step).

Convergence idea

Assume that f is strongly convex, such that $mI \preceq \nabla^2 f(x) \preceq MI$, and that the Hessian $\nabla^2 f$ is L-Lipschitz.
We have the following two phases of convergence:

1. **Damped phase**: far from the optimum, the step $t^{(k)}$ might be less than 1. Each iteration yields an absolute improvement of $-\gamma < 0$.
2. **Quadratic phase**: close to the optimum, the step $t^{(k)} = 1$. Each iteration squares the error.

More precisely, we can show that there exists $0 < \eta \leq m^2/L$ and $\gamma > 0$ such that

- If $\|\nabla f(x^{(k)})\|_2 \geq \eta$, then
$$f(x^{(k+1)}) - f(x^{(k)}) \leq -\gamma$$
- If $\|\nabla f(x^{(k)})\|_2 < \eta$, then $t^{(k)} = 1$ and
$$\frac{L}{2m^2}\|\nabla f(x^{(k+1)})\|_2 \leq \left(\frac{L}{2m^2}\|\nabla f(x^{(k)})\|_2\right)^2$$

Newton is fast around the solution

We have, if $\|\nabla f(x^{(k)})\|_2 < \eta$, then $t^{(k)} = 1$ and
$$\frac{L}{2m^2}\|\nabla f(x^{(k+1)})\|_2 \leq \left(\frac{L}{2m^2}\|\nabla f(x^{(k)})\|_2\right)^2$$
Let $k = k_0 + \ell$, $\ell \geq 1$, with k_0 such that $\|\nabla f(x^{(k_0)})\|_2 < \eta$. Then $\|\nabla f(x^{(k)})\|_2 < \eta$, and,
$$\frac{L}{2m^2}\|\nabla f(x^{(k)})\|_2 \leq \left(\frac{L}{2m^2}\|\nabla f(x^{(k-1)})\|_2\right)^2$$
Recursively,
$$\frac{L}{2m^2}\|\nabla f(x^{(k)})\|_2 \leq \left(\frac{L}{2m^2}\|\nabla f(x^{(k_0)})\|_2\right)^{2^\ell} \leq \frac{1}{2^{2^\ell}}$$
And thus
$$f(x^{(k)}) - v^\sharp \leq \frac{1}{2m}\|\nabla f(x^{(k)})\|_2^2 \leq \frac{2m^3}{L^2}\frac{1}{2^{2^{\ell-1}}}$$
$\rightsquigarrow$ in the quadratic convergence phase, Newton's algorithm gets the result in a few iterations (5 or 6).

Convergence speed - Wrap-up

The Newton algorithm, for strongly convex function, have two phases :

- The damped phase, where $t^{(k)}$ can be less than 1. Each iteration yields an absolute improvement of $-\gamma < 0$.
- The quadratic phase, where each step $t^{(k)} = 1$.

Thus, the total number of iterations to get an ε solution is bounded above by
$$\frac{f(x^{(0)}) - v^\sharp}{\gamma} + \underbrace{\log_2(\log_2(\varepsilon_0/\varepsilon))}_{\lesssim 6}$$
where $\varepsilon_0 = 2m^3/L^2$.

Note that, in 6 iterations in the quadratic convergent phase we get an error $\varepsilon \approx 5.10^{-20}\varepsilon_0$.

Newton's properties in a nutshell

- Full Newton step : $x^{(k+1)} = x^{(k)} - [\nabla^2 f(x^{(k)})]^{-1}\nabla f(x^{(k)})$
- Can be seen through various lenses:
  1. $[\nabla^2 f(x^{(k)})]^{-1}\nabla f(x^{(k)})$ is a descent direction (f is strongly convex);
  2. model-based algorithm where the model is the second-order approximation;
  3. preconditioned gradient algorithm, with adaptive preconditioning.
- Is incredibly fast around the optimal solution.
- Far from the optimum a full Newton step is a bad idea:
  - If f is not strongly convex the Newton direction might not be a descent direction[2]!
  - $\rightsquigarrow$ check if it is a descent direction, otherwise make a gradient step.
  - Even with convexity the step might be too aggressive, $\rightsquigarrow$ receeding step choice.
- Convergence of the (damped) Newton's algorithm is in two phases:
  - slow constant update far from the optimum,
  - fast updates with full step close to the optimum.

[2] It can, for example, get you to the maximum of the second-order approximation...

The main idea ♡

Newton's step is very efficient (near optimality) but has three drawbacks:

1. having a second-order oracle to compute the Hessian
2. storing the Hessian (n^2 values)
3. solving a (dense) linear system : $\nabla^2 f(x^{(k)})d = -\nabla f(x^{(k)})$

The main idea of Quasi Newton method is to define $M^{(k)} \approx \nabla^2 f(x^{(k)})$ (or $W^{(k)} \approx [\nabla^2 f(x^{(k)})]^{-1}$):

1. from first order information $\rightsquigarrow$ no need to compute Hessian;
2. sparse $\rightsquigarrow$ smaller storage requirements;
3. $d^{(k)} = -W^{(k)}\nabla f(x^{(k)}) \rightsquigarrow$ no linear system solving.

Conditions on the approximate Hessian I ♢

We want to construct $M^{(k)}$ an approximation of $\nabla^2 f(x^{(k)})$, leading to a quadratic model of f at iteration k

$$f^{(k)}(x) := f(x^{(k)}) + \left\langle \nabla f(x^{(k)}), x - x^{(k)} \right\rangle + \frac{1}{2}(x - x^{(k)})^\top M^{(k)}(x - x^{(k)})$$

We ask that the gradient of the model $f^{(k)}$ and the true function to match at the current and last iterates:

$$\begin{cases} \nabla f^{(k)}(x^{(k)}) = \nabla f(x^{(k)}) \\ \nabla f^{(k)}(x^{(k-1)}) = \nabla f(x^{(k-1)}) \end{cases}$$

This simply write as the Quasi-Newton equation

$$M^{(k)} \underbrace{(x^{(k)} - x^{(k-1)})}_{\delta_x^{(k-1)}} = \underbrace{\nabla f(x^{(k)}) - \nabla f(x^{(k-1)})}_{\delta_g^{(k-1)}}$$

♣ Exercise: prove it

Conditions on the approximate Hessian II ♢

We are looking for a matrix M such that

- $M \succ 0$
- $M\delta_x = \delta_g$ (only possible if $\delta_g^\top \delta_x > 0$ ♣ Exercise: prove it)
- $M^\top = M$
- M is constructed from first order information only
- If possible, M is sparse

$\rightsquigarrow$ an infinite number of solutions as we have $n(n+1)/2$ variables and n constraints.

$\rightsquigarrow$ Numerous quasi-Newton algorithms developed and tested between 1960-1980.

Choosing the approximate Hessian $M^{(k)}$ ♢

At the end of iteration k we have determined

- $x^{(k+1)}$ and $\delta_x^{(k)} = x^{(k+1)} - x^{(k)}$
- $g^{(k+1)} = \nabla f(x^{(k+1)})$ and $\delta_g^{(k)} = g^{(k+1)} - g^{(k)}$

and we are looking for $M^{(k+1)} \approx \nabla^2 f(x^{(k+1)})$ satisfying the previous requirement.

The idea is to choose $M^{(k+1)}$ close to $M^{(k)}$, that is to solve (analytically)

$$\begin{aligned} \underset{M \in S_{++}^n}{\text{Min}} \quad & d(M, M^{(k)}) \\ s.t. \quad & M\delta_x^{(k)} = \delta_g^{(k)} \end{aligned}$$

for some distance d.

BFGS

Broyden-Fletcher-Goldfarb-Shanno chose

$$d(A,B) := \mathrm{tr}(AB) - \ln\det(AB)$$

A few remarks

- $\Psi : M \mapsto \mathrm{tr}\, M - \ln\det(M)$ is convex on S_{++}^n
- For $M \in S_{++}^n$, $\mathrm{tr}\, M - \ln\det(M) = \sum_{i=1}^n \lambda_i - \ln(\lambda_i)$
- Ψ is minimized in the identity matrix
- $d(A,B) - n$ is the Kullback-Lieber divergence between $\mathcal{N}(0,A)$ and $\mathcal{N}(0,B)$

BFGS update

One of the pragmatic reasons for this choice of distance is that the optimal solution can be found analytically.
We have[3] (to alleviate notation we drop the index k on $\delta_x^{(k)}$ and $\delta_g^{(k)}$)

$$M^{(k+1)} = M^{(k)} + \frac{\delta_g \delta_g^\top}{\delta_g^\top \delta_x} - \frac{M^{(k)} \delta_x \delta_x^\top M^{(k)}}{\delta_x^\top M^{(k)} \delta_x}$$

Even better, denoting $W = M^{-1}$, we can show[4] that:

$$W^{(k+1)} = \left(I - \frac{\delta_x \delta_g^\top}{\delta_g^\top \delta_x}\right) W^{(k)} \left(I - \frac{\delta_g \delta_x^\top}{\delta_g^\top \delta_x}\right) + \frac{\delta_x \delta_x^\top}{\delta_g^\top \delta_x}$$

[3] with some effort
[4] fastidiously

BFGS algorithm

Data: Initial point $x^{(0)}$, First order oracle, error $\varepsilon > 0$.
$W^{(0)} = I$;
while $\|\nabla f(x^{(k)})\| \geq \varepsilon$ **do**
  $g^{(k)} := \nabla f(x^{(k)})$;
  $d^{(k)} := -W^{(k)} g^{(k)}$;
  Compute $t^{(k)}$ by backtracking line-search, starting from $t = 1$;
  $x^{(k+1)} = x^{(k)} + t^{(k)} d^{(k)}$;
  $\delta_g = g^{(k+1)} - g^{(k)}$, $\delta_x = x^{(k+1)} - x^{(k)}$;
  $W^{(k+1)} = \left(I - \frac{\delta_x \delta_g^\top}{\delta_g^\top \delta_x}\right) W^{(k)} \left(I - \frac{\delta_g \delta_x^\top}{\delta_g^\top \delta_x}\right) + \frac{\delta_x \delta_x^\top}{\delta_g^\top \delta_x}$;
  $k = k + 1$;

Algorithm 2: BFGS algorithm

- ✔ First order oracle only
- ✔ No need to solve a linear system
- ✖ Still large memory requirement
- ✔ Convergence comparable to Newton's algorithm

Limited-memory BFGS (L-BFGS)

- For $n \geq 10^3$ storing the matrices is a difficulty.
- Instead of storing and updating the matrix $W^{(k)}$ we store (δ_x, δ_g) pairs.
- We can then compute $d^{(k)} = -W^{(k)} g^{(k)}$ directly from the last 5 to 20 pairs, using recursively the update rule and never computing $W^{(k)}$.

$\rightsquigarrow$ An algorithm with:

- ✔ First order oracle only
- ✔ No need to solve a linear system
- ✔ Same storage requirement as gradient algorithm
- ✔ Convergence comparable to Newton's algorithm

$\rightsquigarrow$ this is the "go to" algorithm when you want high-level precision for strongly convex smooth problems. It is the default choice in a lot of optimization libraries.

What you have to know

- At least one idea behind Newton's algorithm.
- The Newton step.
- That quasi-Newton methods are almost as good as Newton, without requiring a second order oracle.

What you really should know

- Newton's algorithm default step is 1, but you should use backtracking step anyway.
- Newton's algorithm converges in two phases : a slow damped phase, and a very fast quadratically convergent phase close to the optimum (at most 6 iterations).
- BFGS is the by default quasi-Newton method. It work by updating an approximation of the inverse of the Hessian close to the precedent approximation and satisfying some natural requirement.
- L-BFGS limit the memory requirement by never storing the matrix but only the step and gradient updates.

What you have to be able to do

- Implement a damped Newton method.

What you should be able to do

- Implement a BFGS method (with the update formula in front of your eyes)

Constrained optimization

V. Leclère (ENPC)

May 22nd, 2026

Why should I bother to learn this stuff?

- Most real problems have constraints that you have to deal with.
- This course give a snapshot of the tools available to you.
- $\Longrightarrow$ useful for
  - having an idea of what can be done when you have constraints

Constrained optimization problem

- In the previous courses we have developed algorithms for unconstrained optimization problem.
- We now want to sketch some methods to deal with the constrained problem

$$\begin{aligned} \operatorname*{Min}_{x\in\mathbb{R}^n} \quad & f(x) \\ \text{s.t.} \quad & x \in X \end{aligned}$$

- We are going to discuss multiple types of constraints set X:
  - X is a ball : $\{x \mid \|x - x_0\|_2 \leq r\}$
  - X is a box : $\{x \mid \underline{x_i} \leq x_i \leq \bar{x_i} \quad \forall i \in [n]\}$
  - X is a polyhedron: $\{x \mid Ax \leq b\}$
  - X is given through explicit constraints $\{x \mid g(x) = 0, \quad h(x) \leq 0\}$

How to deal with constraints: a method map ♡

We want to solve

$$\min_{x\in X} f(x),$$

where X may be simple, polyhedral, or defined by general constraints.

Constraint structure	**Main idea**	**Typical methods**
X *simple* (ball, box, easy cone)	Project after each step	Projected gradient, proximal methods
X *polyhedral / simplex*	Minimize linearization over X	Conditional gradient (Frank–Wolfe)
General $g(x) = 0$, $h(x) \leq 0$	Move constraints into the cost	Quadratic / L^1 penalization, augmented Lagrangian
General, separable structure	Dualize coupling constraints	Dual ascent, Uzawa, decomposition

Admissible descent direction

- Recall that a descent direction d at point $x^{(k)} \in \mathbb{R}^n$ is a vector such that $\nabla f(x^{(k)})^\top d < 0$.
- An admissible descent direction at point $x^{(k)} \in X$ is a descent direction $d \in \mathbb{R}^n$ such that,

$$\exists \varepsilon > 0, \quad \forall t \in [0, \varepsilon], \qquad x^{(k)} + td \in X.$$

- In other words, an admissible descent direction, is a direction that locally decreases the objective while staying in the constraint set.
- An admissible descent direction algorithm is naturally defined by:
  - A choice of admissible descent direction $d^{(k)}$
  - A choice of (sufficiently small) step $t^{(k)}$
  - $x^{(k+1)} = x^{(k)} + t^{(k)} d^{(k)} \in X$
- Warning: this does not necessarily converge. We can construct examples where the step size gets increasingly small because of the constraints.

A warning: a naive "masked gradient" can stall ♢

Consider a feasible set $X = \mathbb{R}^n_+$ and a differentiable convex f. A tempting heuristic is to use the masked gradient direction

$$d_i^{(k)} := -\nabla_i f(x^{(k)}) \mathbb{1}_{x_i^{(k)} > 0},$$

i.e. we freeze coordinates that are currently on the boundary.

- This direction is *feasible* for the orthant (for small $t > 0$).
- But it is *not* a principled way to enforce first-order stationarity on X.
- The correct necessary condition is the normal cone condition

$$0 \in \nabla f(x^\star) + N_X(x^\star) \iff \begin{cases} x^\star \geq 0, \\ \nabla f(x^\star) \geq 0, \\ x_i^\star \nabla_i f(x^\star) = 0 \ \ \forall i, \end{cases}$$

  which the masking rule does not target.

Take-away. "Feasible direction" is not enough. You need a direction rule that is consistent with stationarity (e.g., projection/prox, FW, or a proper active-set / SQP logic).

Conditional gradient algorithm

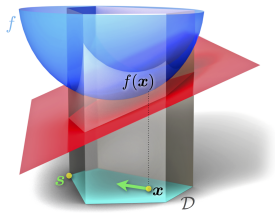
We address an optimization problem with a convex objective function f and compact polyhedral constraint set X, i.e.

$$\min_{x \in X \subset \mathbb{R}^n} \quad f(x)$$

where

$$X = \{x \in \mathbb{R}^n \ \mid \ Ax \leq b, \quad \tilde{A}x = \tilde{b}\}$$

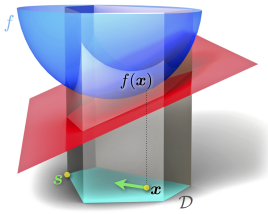
It is a descent algorithm, where we first look for an admissible descent direction $d^{(k)}$, and then look for the optimal step.



Conditional gradient algorithm

The conditional gradient method consists in choosing the descent direction that minimizes the linearization of f over X. More precisely, at step k we solve

$$y^{(k)} \in \arg\min_{y \in X} \quad f(x^{(k)}) + \nabla f(x^{(k)}) \cdot (y - x^{(k)}).$$



Conditional gradient (Frank–Wolfe): update + certificate ♡

Given $x^{(k)} \in X$, compute a linear minimization oracle

$$y^{(k)} \in \arg\min_{y \in X} \ \nabla f(x^{(k)})^\top y, \qquad d^{(k)} := y^{(k)} - x^{(k)}.$$

- **Update (always feasible):**

$$x^{(k+1)} = (1-\gamma_k)x^{(k)} + \gamma_k y^{(k)}, \qquad \gamma_k \in [0,1]$$

  (via line search or a fixed rule, e.g. $\gamma_k = \frac{2}{k+2}$ for smooth convex).
- **Frank–Wolfe gap (optimality certificate):**

$$g_{\mathrm{FW}}(x^{(k)}) := \nabla f(x^{(k)})^\top \left(x^{(k)} - y^{(k)}\right) \ \geq 0.$$

  If f is convex, then $f(x^{(k)}) - f^\star \leq g_{\mathrm{FW}}(x^{(k)})$ and $g_{\mathrm{FW}}(x^{(k)}) = 0 \Leftrightarrow x^{(k)}$ optimal.

Remarks on conditional gradient

$$y^{(k)} \in \arg\min_{y \in X} \quad f(x^{(k)}) + \nabla f(x^{(k)}) \cdot (y - x^{(k)}).$$

- This problem is linear, hence easy to solve.
- By the convexity inequality, the value of the linearized Problem is a lower bound to the true problem.
- As $y^{(k)} \in X$, $d^{(k)} = y^{(k)} - x^{(k)}$ is a *feasable direction*, in the sense that for all $t \in [0,1]$, $x^{(k)} + t d^{(k)} \in X$.
- If $y^{(k)}$ is obtained through the simplex method it is an extreme point of X, which means that, for $t > 1$, $x^{(k)} + t d^{(k)} \notin X$.
- If $y^{(k)} = x^{(k)}$ then we have found an optimal solution.
- We also have $y^{(k)} \in \arg\min_{y \in X} \nabla f(x^{(k)}) \cdot y$, the lower-bound being obtained easily.

Projection on a convex set ♡

Let $X \subset \mathbb{R}^n$ be a nonempty closed convex set. We call $P_X : \mathbb{R}^n \to \mathbb{R}^n$ the projection on X the fonction such that

$$P_X(x) = \arg\min_{x' \in X} \|x' - x\|_2^2$$

We have

- $\bar{x} = P_X(x)$ iff $(x - \bar{x}) \in N_X(\bar{x})$ (i.e. $\langle x - \bar{x}\,, x' - \bar{x} \rangle \leq 0, \quad \forall x' \in X$)
- $\langle P_X(y) - P_X(x)\,, y - x \rangle \geq 0$ (P_X is *non-decreasing*)
- $\|P_X(y) - P_X(x)\|_2 \leq \|y - x\|$ (P_X is a *contraction*)

♠ Exercise: Prove these results

Projected gradient I ♡

Consider

$$\begin{aligned} \min_{x \in \mathbb{R}^n} \quad & f(x) \\ \text{s.t.} \quad & x \in X \end{aligned}$$

where f is differentiable and X convex.
The projected gradient algorithm generates the following sequence

$$x^{(k+1)} = P_X\left[x^{(k)} - t^{(k)} g^{(k)}\right]$$

Projected gradient: stationarity + convergence statement

II ♢

Let $X \subset \mathbb{R}^n$ be nonempty closed convex and f differentiable.

Stationarity on X

$x^\sharp \in X$ is (first-order) stationary iff

$$0 \in \nabla f(x^\sharp) + N_X(x^\sharp) \iff x^\sharp = P_X\big(x^\sharp - t\nabla f(x^\sharp)\big) \text{ for some (equiv. all) } t > 0.$$

Projected gradient mapping (certificate)

$$G_t(x) := \frac{1}{t}\Big(x - P_X(x - t\nabla f(x))\Big), \qquad G_t(x) = 0 \iff x \text{ stationary.}$$

If f has L-Lipschitz gradient and $t \in (0, 2/L)$ is fixed, projected gradient satisfies $f(x^{(k)})$ decreases and $\|G_t(x^{(k)})\| \to 0$. If f is convex, every cluster point is optimal; if the minimizer is unique (e.g. f strongly convex), then $x^{(k)} \to x^\star$.

When to use ?

♡

- Projected gradient is useful only if the projection is simple, as projecting over a convex set consists in solving a constrained optimization problem.
- Projection is simple for balls and boxes.
- Finding an admissible direction is doable if the constraint set is polyhedral, or more generally conic-representable.

Idea of penalization

♡

We consider the constrained optimization problem

$$(\mathcal{P}) \quad \begin{array}{ll} \underset{x \in \mathbb{R}^n}{\text{Min}} & f(x) \\ \text{s.t.} & x \in X \end{array}$$

and the following penalized version

$$(\mathcal{P}_r) \quad \underset{x \in \mathbb{R}^n}{\text{Min}} \quad f(x) + r p(x)$$

Thus, a (constrained) problem is replaced by a sequence of (unconstrained) problems.
♣ Exercise: What is happening if $p = \mathbb{I}_X$?

Some monotonicity results

♢

$$(\mathcal{P}_r) \quad \underset{x \in \mathbb{R}^n}{\text{Min}} \quad f(x) + r p(x)$$

The idea is that, with higher r, the penalization has more impact on the problem.
More precisely, let $0 < r_1 < r_2$, and x_{r_i} be an optimal solution of $(\mathcal{P}_{r_i})$.
We have:

- $p(x_{r_1}) \geq p(x_{r_2})$
- $f(x_{r_1}) \leq f(x_{r_2})$

♣ Exercise: prove these results.

Outer penalization

A first idea for choosing a penalization function p consists in choosing a function p such that:

- $p(x) = 0$ for $x \in X$
- $p(x) > 0$ for $x \notin X$

intuitively the idea is that p is the fine to pay for not respecting the constraint. Heuristically, it should be increasing with the distance to X.

Outer penalization - theoretical results

Assume that

- p is l.s.c on $\mathbb{R}^n$
- $p \geq 0$
- $p(x) = 0$ iff $x \in X$

Further assume that f is l.s.c and there exists $r_0 > 0$ such that $x \mapsto f(x) + r_0 p(x)$ is coercive (i.e. $\to \infty$ if $\|x\| \to \infty$).
Then,

1. for $r > r_0$, $(\mathcal{P}_r)$ admit at least one optimal solution
2. $(x_r)_{r \to +\infty}$ is bounded
3. any adherence point of $(x_r)_{r \to +\infty}$ is an optimal solution of $\mathcal{P}$.

Outer penalization - quadratic case

Assume that

$$X = \{x \in \mathbb{R}^n \quad | \quad g(x) = 0, \quad h(x) \leq 0\}$$

then the quadratic penalization consists in choosing

$$p : x \mapsto \|g(x)\|^2 + \|(h(x))^+\|^2$$

This choice is interesting as (for affinely lower-bounded f):

- if f, g, h are C^1, then $x \mapsto f(x) + rp(x)$ is C^1
- as $r \to \infty$, any cluster point of (x_r) is feasible and (under standard assumptions) optimal for $(\mathcal{P})$

However, generally speaking, if the constraints are impactful (e.g. have non-zero optimal multipliers), then

$$x_r \notin X$$

Outer penalization - L^1 case

Assume that

$$X = \{x \in \mathbb{R}^n \quad | \quad g(x) = 0, \quad h(x) \leq 0\}$$

another natural penalization consists in choosing

$$p : x \mapsto \|g(x)\|_1 + \|(h(x))^+\|_1$$

The interest of this approach is that, if the problem is convex and the constraints are qualified at optimality, then, for r large enough, an optimal solution to the penalized problem $(\mathcal{P}_r)$ is an optimal solution to the original problem $(\mathcal{P})$. Thus, we speak of exact penalization.

Unfortunately, this comes to the price of non-differentiability.

Inner penalization

Another approach consists in choosing a penalization function p that takes value $+\infty$ outside of X.

The idea here is to add a potential that keeps the *iterates* away from the boundary (while approaching optimality as the barrier vanishes).

This is typically done in a way to keep $f + \frac{1}{s}p$ smooth, and if possible convex.

Note that, for the inner penalization, we need the coefficient $\frac{1}{s} \to 0$, (hence $s \to +\infty$) for the penalized problem to converges toward the original one.

More on that in the next course.

Duality, here we go again ♡

Recall that to a primal problem

$$(\mathcal{P}) \quad \min_{x \in \mathbb{R}^n} \quad f(x) \tag{1}$$
$$\text{s.t.} \quad g(x) = 0 \tag{2}$$
$$h(x) \leq 0 \tag{3}$$

we associate the dual problem

$$(\mathcal{D}) \quad \max_{\lambda, \mu \geq 0} \quad \underbrace{\min_x \; f(x) + \lambda^\top g(x) + \mu^\top h(x)}_{\Phi(\lambda, \mu)}$$

♣ Exercise: Under which sufficient conditions are these problems equivalent ?

Duality as exact penalization (via a saddle point) ♡

Consider the convex problem

$$\min_x \; f(x) \quad \text{s.t. } g(x) = 0, \; h(x) \leq 0, \qquad \text{with Slater (so strong duality + multipliers).}$$

Let $(x^\star, \lambda^\star, \mu^\star)$ be a saddle point of the Lagrangian $L(x, \lambda, \mu) = f(x) + \lambda^\top g(x) + \mu^\top h(x)$ with $\mu^\star \geq 0$. Then

$$x^\star \in \arg\min_x L(x, \lambda^\star, \mu^\star),$$

and $x^\star$ is primal optimal.

Interpretation. With the *right* multipliers, $L(\cdot, \lambda^\star, \mu^\star)$ acts like an exact penalization of the constraints.

Dual ascent as projected (sub)gradient

Dual function:

$$\Phi(\lambda, \mu) := \inf_x \; \big(f(x) + \lambda^\top g(x) + \mu^\top h(x)\big), \qquad \mu \geq 0,$$

is concave and typically nonsmooth.

If $x^\sharp(\lambda, \mu) \in \arg\min_x L(x, \lambda, \mu)$, then (Danskin)

$$\begin{pmatrix} g(x^\sharp(\lambda, \mu)) \\ h(x^\sharp(\lambda, \mu)) \end{pmatrix} \in \partial \Phi(\lambda, \mu).$$

Projected subgradient ascent with step $t > 0$:

$$\lambda^{(k+1)} = \lambda^{(k)} + t\, g(x^{(k+1)}), \qquad \mu^{(k+1)} = \big[\mu^{(k)} + t\, h(x^{(k+1)})\big]^+.$$

Uzawa's algorithm

Data: Initial primal point $x^{(0)}$, Initial dual points $\lambda^{(0)}, \mu^{(0)}$, unconstrained optimization method, dual step $t > 0$.
while $\|g(x^{(k)})\|_2 + \|(h(x^{(k)}))^+\|_2 \geq \varepsilon$ **do**
  Solve for $x^{(k+1)}$

$$\min_x \quad f(x) + \lambda^{(k)\top} g(x) + \mu^{(k)\top} h(x)$$

  Update the multipliers

$$\lambda^{(k+1)} = \lambda^{(k)} + t g(x^{(k+1)})$$
$$\mu^{(k+1)} = [\mu^{(k)} + t h(x^{(k+1)})]^+$$

Algorithm 1: Uzawa algorithm

Convergence requires strong convexity and constraint qualifications.

Exercise: decomposition by prices

We consider the following energy problem:

- you are an energy producer with N production unit
- you have to satisfy a given demand planning for the next 24h (i.e. the total output at time t should be equal to d_t)
- the time step is the hour, and each unit has a production cost for each planning given as a convex quadratic function of the planning

1. Model this problem as an optimization problem. In which class does it belong? How many variables?
2. Apply Uzawa's algorithm to this problem. Why could this be an interesting idea?
3. Give an economic interpretation of this method.
4. What would happen if each unit had production constraints?

What you have to know

- There is four main ways of dealing with constraints:
  - choosing an admissible direction
  - projection of the next iterate
  - penalizing the constraints
  - dualizing the constraints

What you really should know

- admissible direction methods are mainly useful for polyhedral constraint set
- projection is useful only if the admissible set is simple (ball or bound constraints)
- penalization can be inner or outer, differentiable or not.

What you have to be able to do

- Implement a penalization approach.

What you should be able to do

- Implement Uzawa's algorithm.

Interior Points Methods

V. Leclère (ENPC)

May 29th, 2026

Why should I bother to learn this stuff?

- Interior point methods are competitive with simplex methods for linear programs
- Interior point methods are state of the art for most conic (convex) problems (inc. SOCP, SDP, etc.)
- $\Longrightarrow$ useful for
  - understanding what is used in numerical solvers (like MOSEK, Gurobi, SCS, etc.)
  - specialization in optimization

Convex differentiable optimization problem

We consider the following convex optimization problem

$$\begin{aligned}(\mathcal{P}) \quad \min_{x\in\mathbb{R}^n} \quad & f(x) \\ \text{s.t.} \quad & Ax = b \\ & h_i(x) \leq 0 \qquad \forall i \in [n_I]\end{aligned}$$

where A is a $n_E \times n$ matrix, and all functions f and h_i are assumed convex, real valued and twice differentiable.

Introducing the Lagrangian

I ♡

$$\begin{aligned}(\mathcal{P}) \quad \min_{x\in\mathbb{R}^n} \quad & f(x) \\ \text{s.t.} \quad & Ax = b \\ & h_i(x) \leq 0 \qquad \forall i \in [n_I]\end{aligned}$$

is equivalent to

$$\min_{x\in\mathbb{R}^n} \quad f(x) + \mathbb{I}_{\{Ax-b=0\}} + \sum_{i=1}^{n_I} \mathbb{I}_{\{h_i(x)\leq 0\}}$$

which we rewrite

$$\min_{x\in\mathbb{R}^n} \sup_{\lambda\in\mathbb{R}^{n_E},\mu\in\mathbb{R}^{n_I}_+} \quad f(x) + \sup_{\lambda\in\mathbb{R}^{n_E}} \lambda^\top(Ax-b) + \sum_{i=1}^{n_I} \sup_{\mu_i\geq 0} \mu_i h_i(x)$$

Introducing the Lagrangian II ♡

$$(\mathcal{P}) \quad \min_{x\in\mathbb{R}^n} \sup_{\lambda\in\mathbb{R}^{n_E},\mu\in\mathbb{R}^{n_I}_+} \underbrace{f(x) + \lambda^\top(Ax-b) + \sum_{i=1}^{n_I} \mu_i h_i(x)}_{:=\mathcal{L}(x;\lambda,\mu)}$$

$$(\mathcal{D}) \quad \sup_{\lambda\in\mathbb{R}^{n_E},\mu\in\mathbb{R}^{n_I}_+} \min_{x\in\mathbb{R}^n} \quad f(x) + \lambda^\top(Ax-b) + \sum_{i=1}^{n_I} \mu_i h_i(x)$$

As for any function ϕ we always have

$$\sup_y \inf_x \phi(x,y) \leq \inf_x \sup_y \phi(x,y)$$

we have that (weak duality)

$$val(\mathcal{D}) \leq val(\mathcal{P}).$$

Lower bounds from duality ♡

Define the dual function

$$d(\lambda,\mu) := \inf_x \mathcal{L}(x;\lambda,\mu)$$

Then we have $val(\mathcal{D}) = \sup_{\lambda\in\mathbb{R}^{n_E},\mu\in\mathbb{R}^{n_I}_+} d(\lambda,\mu)$.

Thus, we can compute a lower bound to $val(\mathcal{D}) \leq val(\mathcal{P})$ by choosing any admissible dual points $\lambda \in \mathbb{R}^{n_E}, \mu \in \mathbb{R}^{n_I}_+$ and solving the unconstrained problem

$$d(\lambda,\mu) = \inf_{x\in\mathbb{R}^n} f(x) + \lambda^\top(Ax-b) + \sum_{i=1}^{n_I} \mu_i h_i(x)$$

Constraint qualification

Recall that, for a convex differentiable optimization problem, the constraints are qualified if *Slater's condition* is satisfied :

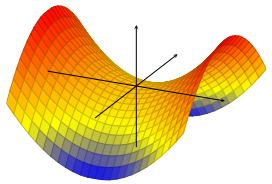
$$\exists x_0 \in \mathbb{R}^n, \qquad Ax_0 = b, \quad \forall i \in [n_I], \quad h_i(x_0) < 0$$

i.e., there exists a strictly admissible feasible point

Saddle point ♡

If $(\mathcal{P})$ is a convex optimization problem with qualified constraints, then

- $val(\mathcal{D}) = val(\mathcal{P})$
- any optimal solution $x^\sharp$ of $(\mathcal{P})$ is part of a saddle point $(x^\sharp;\lambda^\sharp,\mu^\sharp)$ of $\mathcal{L}$
- $(\lambda^\sharp,\mu^\sharp)$ is an optimal solution of $(\mathcal{D})$



Karush Kuhn Tucker conditions ♡

If Slater's condition is satisfied, then $x^\sharp$ is an optimal solution to (P) if and only if there exists optimal multipliers $\lambda^\sharp \in \mathbb{R}^{n_E}$ and $\mu^\sharp \in \mathbb{R}^{n_I}$ satisfying

$$\begin{cases} \nabla f(x^\sharp) + A^\top \lambda^\sharp + \sum_{i=1}^{n_I} \mu_i^\sharp \nabla h_i(x^\sharp) = 0 & \text{first order condition} \\ Ax^\sharp = b & \text{primal admissibility} \\ h(x^\sharp) \leq 0 & \\ \mu^\sharp \geq 0 & \text{dual admissibility} \\ \mu_i^\sharp h_i(x^\sharp) = 0, \quad \forall i \in [n_I] & \text{complementarity} \end{cases}$$

The three last conditions are sometimes compactly written

$$0 \geq h(x^\sharp) \perp \mu \geq 0$$

Intuition for Newton's method: unconstrained case ♡

Newton's method is an iterative optimization method that minimizes a quadratic approximation of the objective function at the current point $x^{(k)}$.
Consider the following unconstrained optimization problem (f smooth):

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

At $x^{(k)}$ we have

$$f(x^{(k)} + d) = f(x^{(k)}) + \nabla f(x^{(k)})^\top d + \frac{1}{2} d^\top \nabla^2 f(x^{(k)}) d + o(\|d\|^2)$$

And the direction $d^{(k)}$ minimizing the quadratic approximation is given by solving for d

$$\nabla f(x^{(k)}) + \nabla^2 f(x^{(k)}) d = 0.$$

Intuition for Newton's method: constrained case ♡

Approximate the linearly constrained optimization problem

$$\begin{aligned} \min_{x \in \mathbb{R}^n} \quad & f(x) \\ \text{s.t.} \quad & Ax = b \end{aligned}$$

by

$$\begin{aligned} \min_{d \in \mathbb{R}^n} \quad & f(x^{(k)}) + \nabla f(x^{(k)})^\top d + \frac{1}{2} d^\top \nabla^2 f(x^{(k)}) d \\ \text{s.t.} \quad & A(x^{(k)} + d) = b \end{aligned}$$

Which is equivalent to solving (for given admissible $x^{(k)}$)

$$\begin{aligned} \min_{d \in \mathbb{R}^n} \quad & \nabla f(x^{(k)})^\top d + \frac{1}{2} d^\top \nabla^2 f(x^{(k)}) d \\ \text{s.t.} \quad & Ad = 0 \end{aligned}$$

Finding Newton's direction

$$\begin{aligned} \min_{d \in \mathbb{R}^n} \quad & \nabla f(x^{(k)})^\top d + \frac{1}{2} d^\top \nabla^2 f(x^{(k)}) d \\ \text{s.t.} \quad & Ad = 0 \end{aligned}$$

By KKT the optimal $d^{(k)}$ is given by solving for (d, λ)

$$\begin{cases} \nabla f(x^{(k)}) + \nabla^2 f(x^{(k)}) d + A^\top \lambda = 0 \\ Ad = 0 \end{cases}$$

Or in a matricial form

$$\begin{pmatrix} \nabla^2 f(x^{(k)}) & A^\top \\ A & 0 \end{pmatrix} \begin{pmatrix} d \\ \lambda \end{pmatrix} = \begin{pmatrix} -\nabla f(x^{(k)}) \\ 0 \end{pmatrix}$$

Newton's algorithm: equality constrained case

Data: Initial admissible point x_0
Result: quasi-optimal point
$k = 0$;
while $\|\nabla f(x^{(k)}) + A^\top \lambda\| \geq \varepsilon$ **do**
  Solve for d

$$\begin{pmatrix} \nabla^2 f(x^{(k)}) & A^\top \\ A & 0 \end{pmatrix} \begin{pmatrix} d \\ \lambda \end{pmatrix} = \begin{pmatrix} -\nabla f(x^{(k)}) \\ 0 \end{pmatrix}$$

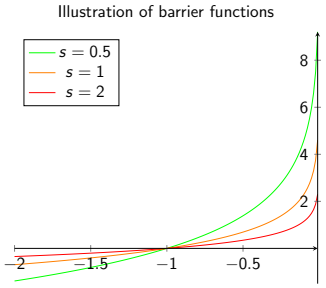
  Line-search for $\alpha \in [0,1]$ on $f(x^{(k)} + \alpha d^{(k)})$
  $x^{(k+1)} = x^{(k)} + \alpha d^{(k)}$
  $k \leftarrow k + 1$

Algorithm 1: Newton's algorithm

The negative log function

- The idea of barrier method is to replace the indicator function $\mathbb{I}_{\mathbb{R}^-}$ by a smooth function.
- We choose the function $z \mapsto -1/s \log(-z)$
- Note that they also take value $+\infty$ on $\mathbb{R}^+$

➥ In particular these barrier functions ensure strict feasibility of the iterates.



Constrained optimization problem

We now want to consider a convex differentiable optimization problem with equality and inequality constraints.

$$\begin{aligned} (\mathcal{P}_\infty) \qquad \min_{x \in \mathbb{R}^n} \quad & f(x) \\ \text{s.t.} \quad & Ax = b \\ & h_i(x) \leq 0 \qquad \forall i \in [n_I] \end{aligned}$$

where all functions f and h_i are assumed convex, finite valued and twice differentiable. Which we rewrite

$$\begin{aligned} \min_{x \in \mathbb{R}^n} \quad & f(x) + \sum_{i=1}^{n_I} \mathbb{I}_{\mathbb{R}^-}(h_i(x)) \\ \text{s.t.} \quad & Ax = b \end{aligned}$$

Calculus ♢

- We define

$$\phi(x) := \begin{cases} -\sum_{i=1}^{n_I} \ln\big(-h_i(x)\big), & \text{if } h_i(x) < 0 \ \forall i \in [n_I], \\ +\infty, & \text{otherwise.} \end{cases}$$

- Thus we have $\frac{1}{s}\phi(x) \xrightarrow[s \to +\infty]{} \mathbb{I}_{\{h_i(x) < 0,\ \forall i \in [n_I]\}}$
- On the domain $\{x : h_i(x) < 0\}$, we have:

$$\nabla \phi(x) = \sum_{i=1}^{n_I} -\frac{1}{h_i(x)} \nabla h_i(x)$$

$$\nabla^2 \phi(x) = \sum_{i=1}^{n_I} \left[\frac{1}{h_i^2(x)} \nabla h_i(x) \nabla h_i(x)^\top - \frac{1}{h_i(x)} \nabla^2 h_i(x) \right]$$

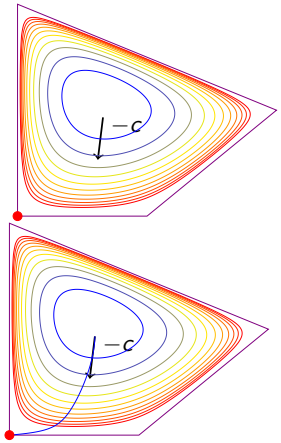
Penalized problem ♡

We consider

$$(\mathcal{P}_{\infty s}) \quad \min_{x\in\mathbb{R}^n} sf(x) + \frac{1}{s}\phi(x) \quad \text{s.t.} \quad Ax = b$$

with optimal solution $x_s^\sharp$.

scaling objective function by s does not change the solution.
Letting s goes to $+\infty$ make x_s go to the solution of $(\mathcal{P})$ along the central path.



Characterizing central path I ♢

x_s is solution of

$$(\mathcal{P}_s) \quad \min_{x\in\mathbb{R}^n} sf(x) + \phi(x) \quad \text{s.t.} \quad Ax = b$$

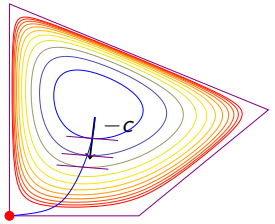
if and only if, there exists $\lambda_s \in \mathbb{R}^{n_E}$, such that

$$\begin{cases} Ax_s = b \\ h_i(x_s) < 0 & \forall i \in [n_I] \\ s\nabla f(x_s) + \nabla\phi(x_s) + A^\top\lambda = 0 \end{cases}$$

Characterizing central path II ♢

$$\begin{cases} Ax_s = b \\ h(x_s) < 0 \\ s\nabla f(x_s) + \nabla\phi(x_s) + A^\top\lambda = 0 \end{cases}$$

If $A = 0$ it means that $\nabla f(x_s)$ is orthogonal to the level lines of ϕ



Duality ♢

Recall the original optimization problem

$$(\mathcal{P}_\infty) \quad \min_{x\in\mathbb{R}^n} f(x) \quad \text{s.t.} \quad Ax = b, \quad h_i(x) \le 0 \quad \forall i \in [n_I]$$

with Lagrangian

$$\mathcal{L}(x;\lambda,\mu) := f(x) + \lambda^\top(Ax - b) + \sum_{i=1}^{n_I} \mu_i h_i(x)$$

and dual function

$$d(\lambda,\mu) := \inf_{x\in\mathbb{R}^n} \mathcal{L}(x;\lambda,\mu).$$

For any admissible dual point $(\lambda,\mu) \in \mathbb{R}^{n_E} \times \mathbb{R}_+^{n_I}$, we have

$$d(\lambda,\mu) \le val(\mathcal{P}_\infty)$$

Getting a lower bound

For given admissible dual point $(\lambda, \mu) \in \mathbb{R}^{n_E} \times \mathbb{R}^{n_I}_+$, a point $x^\sharp(\lambda, \mu)$ minimizing $\mathcal{L}(\cdot, \lambda, \mu)$, is characterized by first order conditions

$$\nabla f(x^\sharp(\lambda,\mu)) + A^\top \lambda + \sum_{i=1}^{n_I} \mu_i \nabla h_i(x^\sharp(\lambda,\mu)) = 0$$

which gives

$$d(\lambda,\mu) = \mathcal{L}(x^\sharp(\lambda,\mu);\lambda,\mu) \leq val(\mathcal{P}_\infty)$$

Dual point on the central path ♢

Now recall that x_s, solution of $(\mathcal{P}_s)$, is characterized by

$$\begin{cases} Ax_s = b, h(x_s) < 0 \\ s\nabla f(x_s) + \nabla\phi(x_s) + A^\top \lambda_s = 0 \end{cases}$$

And we have seen that

$$\nabla\phi(x) = \sum_{i=1}^{n_I} \frac{1}{-h_i(x)} \nabla h_i(x)$$

Thus,

$$\nabla f(x_s) + A^\top \lambda_s/s + \sum_{i=1}^{n_I} \underbrace{\frac{1}{-s h_i(x_s)}}_{(\mu_s)_i} \nabla h_i(x_s) = 0$$

which means that $x_s = x^\sharp(\lambda_s/s, \mu_s)$.

Bounding the error ♢

Let x_s be a primal point on the central path satisfying

$$\exists \lambda_s \in \mathbb{R}^{n_E}, \qquad s\nabla f(x_s) + \nabla\phi(x_s) + A^\top \lambda_s = 0$$

We define a dual point $(\mu_s)_i = \frac{1}{-s h_i(x_s)} > 0$. We have

$$\begin{aligned} d(\mu_s, \lambda_s/s) &= \mathcal{L}(x_s, \mu_s, \lambda_s/s) \\ &= f(x_s) + \frac{1}{s}\lambda_s{}^\top \underbrace{(Ax_s - b)}_{=0} + \sum_{i=1}^{n_I} \frac{1}{-s h_i(x_s)} h_i(x_s) \\ &= f(x_s) - \frac{n_I}{s} \leq val(\mathcal{P}_\infty) \end{aligned}$$

➥ x_s is an n_I/s-optimal solution of $(\mathcal{P}_\infty)$.

Interpretation through KKT condition ♡

A point x_s is on the central path iff it is strictly admissible and there exists $\lambda \in \mathbb{R}^{n_E}$ such that

$$\nabla f(x_s) + A^\top \lambda + \sum_{i=1}^{n_I} \underbrace{\frac{1}{-s h_i(x)}}_{(\mu_s)_i} \nabla h_i(x) = 0$$

which can be rewritten

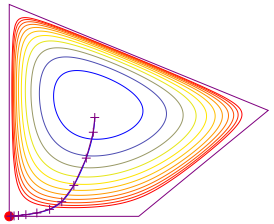
$$\begin{cases} \nabla f(x) + A^\top \lambda + \sum_{i=1}^{n_I} \mu_i \nabla h_i(x) = 0 \\ Ax = b, h_i(x) \leq 0 \\ \mu \geq 0 \\ -\mu_i h_i(x) = \frac{1}{s} \qquad \forall i \in [n_I] \end{cases}$$

Taking a step back

- We saw that we can extend Newton's method to solve linearly constrained optimization problem.
- We saw that we can approximate inequality constraints through the use of logarithmic barrier $-1/s \sum_i \ln(-h_i(x))$.
- We proved that x_s is an n_l/s-optimal solution.
- The trade-off with s is : larger s means x_s closer to optimal solution x_∞ but the approximate problem $(\mathcal{P}_s)$ have worse conditionning.

Barrier method

Data: increase $\rho > 1$, error $\varepsilon > 0$,
initial s
Result: ε-optimal point
solve $(\mathcal{P}_s)$ and set $x = x_s$;
while $n_l/t \geq \varepsilon$ **do**
  increase t: $t = \rho t$ centering step:
  solve $(\mathcal{P}_s)$ starting at x ;
  update : $x = x_s$



Question : why solve $(\mathcal{P}_s)$ to optimality ?

Solving $(\mathcal{P}_s)$ with Newton's method

$$(\mathcal{P}_s) \quad \min_{x \in \mathbb{R}^n} \; sf(x) + \phi(x) \quad \text{s.t.} \quad Ax = b$$

is a linearly constrained optimization problem that can be solved by Newton's method. More precisely we have $x^{(k+1)} = x^{(k)} + d^{(k)}$ with $d^{(k)}$ a solution of

$$\begin{pmatrix} s\nabla^2 f(x^{(k)}) + \nabla^2\phi(x^{(k)}) & A^\top \\ A & 0 \end{pmatrix} \begin{pmatrix} d \\ \lambda \end{pmatrix} = \begin{pmatrix} -s\nabla f(x^{(k)}) - \nabla\phi(x^{(k)}) \\ 0 \end{pmatrix}$$

Path following interior point method

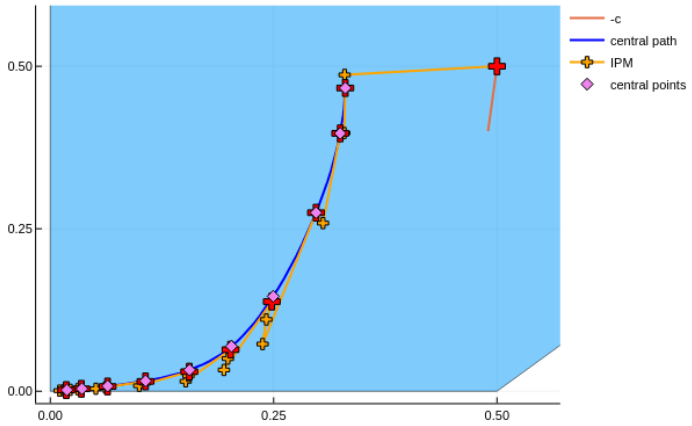
Data: increase $\rho > 1$, error $\varepsilon > 0$, initial s_0
initial strictly feasible point x_0
$k = 0$
$x \leftarrow x_0$, $s \leftarrow s_0$
for $k \in \mathbb{N}$ **do** // Outer step
  for $\kappa \in [K]$ **do** // Inner step
    solve for d ; // Newton step for $(\mathcal{P}_s)$

$$\begin{pmatrix} s_k\nabla^2 f(x) + \nabla^2\phi(x) & A^\top \\ A & 0 \end{pmatrix} \begin{pmatrix} d \\ \lambda \end{pmatrix} = \begin{pmatrix} -s_k\nabla f(x) - \nabla\phi(x) \\ 0 \end{pmatrix}$$

    reduce α from 1 until $f(x + \alpha d) \leq f(x)$;
    $x \leftarrow x + \alpha d$;
  $s \leftarrow \rho s$;

Algorithm 2: Path following algorithm

Path following algorithm



A linear problem - inequality form

We consider the following LP

$$\begin{aligned} \min_{x\in\mathbb{R}^n} \quad & c^\top x \\ \text{s.t.} \quad & a_i^\top x \leq b_i \qquad \forall i \in [n_I] \end{aligned}$$

Where $a_i^\top = A[i,:]$ is the row of matrix A, such that the constraints can be written $Ax \leq b$. Thus, x_s is the solution of

$$\min_{x\in\mathbb{R}^n} \quad sc^\top x + \phi(x)$$

where

$$\phi(x) := -\sum_{i=1}^{n_I} \ln(b_i - a_i^\top x)$$

Calculus ♢

$$\begin{aligned} \phi(x) &= -\sum_{i=1}^{n_I} \ln(b_i - a_i^\top x) \\ \nabla\phi(x) &= \sum_{i=1}^{n_I} \frac{1}{b_i - a_i^\top x} a_i \\ \nabla^2\phi(x) &= \sum_{i=1}^{n_I} \frac{1}{(b_i - a_i^\top x)^2} a_i a_i^\top \end{aligned}$$

This can be written in matrix form, using the vector $d \in \mathbb{R}^{n_I}$ defined by $d_i = \frac{1}{b_i - a_i^\top x}$

$$\begin{aligned} \nabla\phi(x) &= A^\top d \\ \nabla^2\phi(x) &= A^\top diag(d)^2 A \end{aligned}$$

Newton step

Starting from x, the Newton direction for $(\mathcal{P}_s)$ is

$$dir_s(x) = -(\nabla^2\phi(x))^{-1}(sc + \nabla\phi(x))$$

which, in algebraic form, yields

$$dir_s(x) = -[A^\top diag(d)^2 A]^{-1}(sc + A^\top d)$$

with $d_i = 1/(b_i - a_i^\top x)$.

Theory tell us to use a step-size of 1 for Newton's method.

Practice teach us to use a smaller step-size (or linear-search).

Interior Point Method for LP pseudo code

Data: Initial admissible point x_0, initial penalization $s_0 > 0$;
parameter: $\rho > 1$, $N_{in} \geq 1$, $N_{out} \geq 1$;
Result: quasi-optimal point
$x = x0$, $s = s_0$;
for $k = 1..N_{out}$ **do**
  for $\kappa = 1..N_{in}$ **do**
    Compute d, with $d_i = 1/(b_i - a_i^T x)$;
    Solve for dir

$$A^\top \mathrm{diag}(d)^2 A\mathrm{dir} = -(sc + A^\top d)$$

    reduce α from 1 until[a] $f(x + \alpha\mathrm{dir}) \leq f(x)$;
    update $x \leftarrow x + \alpha\mathrm{dir}$;
  update $s \leftarrow \rho s$;

Algorithm 3: Interior Point Method for LP

[a] simplest condition described here

What you have to know

- IPM are state of the art algorithms for LP and more generally conic optimization problem
- That logarithmic barrier are a useful inner penalization method

What you really should know

- That Newton's algorithm can be applied with equality constraints
- What is the central path
- That IPM work with inner and outer optimization loop

Stochastic Gradient Method

Loucas Pillaud-Vivien (ENPC)

June, 2026

Motivation: Finite data set n

Fit $h_\theta(x)$ to learn the input/output function related to $(x_i, y_i)_{1 \leq i \leq n}$.

$$\min_\theta L(\theta) := \frac{1}{n} \sum_{i=1}^{n} \ell(h_\theta(x_i), y_i) \ ,$$

where ℓ is a loss function (e.g. squared loss, logistic loss, etc.).
Gradient descent is too costly as it requires to compute the gradient

$$\nabla L(\theta) = \frac{1}{n} \sum_{i=1}^{n} \nabla_\theta \ell(h_\theta(x_i), y_i) \ .$$

But we can use an estimator of the gradient by picking a random point (x_i, y_i) from the dataset, and compute

$$\hat{g}(\theta) = \nabla_\theta \ell(h_\theta(x_i), y_i) \ .$$

Then we can use this estimator to update the parameter θ as

$$\theta^{(k+1)} = \theta^{(k)} - \alpha \hat{g}(\theta^{(k)}) \ ,$$

where α is a step size.

Population data set: $n = +\infty$. Want to learn a function from a model

$$y = f^*(x) + \sigma \xi \ ,$$

where $x \sim \mathcal{P}$ and $\xi \sim \mathcal{N}(0, Id)$ and have access to infinite pairs (x, y).
One idea is to minimize the population risk

$$\min_\theta L(\theta) = \mathbb{E}\Big[\ell(h_\theta(x), y)\Big] = \int \ell(h_\theta(x), y) dP(x, y) \ .$$

If computing the population risk is not possible, we can use a random sample $(x, y) \sim P(x, y)$ from the population to estimate the gradient

$$\hat{g}(\theta) = \nabla_\theta \ell(h_\theta(x), y) \ .$$

Then we can use this estimator to update the parameter θ as

$$\theta^{(k+1)} = \theta^{(k)} - \alpha \hat{g}(\theta^{(k)}) \ ,$$

where α is a step size.

Why should I bother to learn this stuff?

This is the main algorithm principle for training machine learning model, and in particular deep neural network.

Useful for

- understanding how the library train ML models
- specialization in optimization
- specialization in machine learning

A general succesful paradigm that dates from the 50's that has been used in many fields.

The abstract optimization problem ♡

We consider the following optimization problem

$$\operatorname*{Min}_{\theta \in \mathbb{R}^p} \quad F(\theta) := \mathbb{E}_{\xi}\Big[f(\theta, \xi)\Big]$$

where ξ is a random variable.
Typically,

- ξ is a random vector that represents the law of the data
- We assume we can sample ξ, but not compute the expectation exactly.
- $f(\theta, \xi)$ is a loss function that measures the quality of the model parameterized by θ on the data ξ

Computing the gradient ♡

We want to use **first order methods** to solve the minimization problem

$$F(\theta) := \mathbb{E}\Big[f(\theta, \xi)\Big]$$

Under some regularity conditions (e.g. $f(\cdot, \xi)$ differentiable, $\dfrac{\partial f(\theta, \cdot)}{\partial \theta}$ Lipschitz in θ uniformly in ξ, and ξ integrable) we have

$$\nabla F(\theta) = \mathbb{E}\left[\frac{\partial f}{\partial \theta}(\theta, \xi)\right]$$

This is obvious if ξ is finitely supported : $\operatorname{supp}(\xi) = \{\xi_i\}_{i \in [n]}$, and $\mathbb{P}(\xi = \xi_i) = \frac{1}{n}$,

$$\nabla F(\theta) = \frac{\partial}{\partial \theta}\left(\sum_{i=1}^{n} \frac{1}{n} f(\theta, \xi_i)\right) = \frac{1}{n}\sum_{i=1}^{n} \frac{\partial}{\partial \theta} f(\theta, \xi_i)$$

Standard continuous optimization method

Thus, we are looking at

$$\operatorname*{Min}_{\theta \in \mathbb{R}^p} F(\theta)$$

where F is a convex differentiable function if $f(\cdot, \xi)$ is, and we know how to compute its gradient.

Thus, we should be able to solve our problem through the method presented in earlier courses:

- gradient algorithm
- conjugate gradient
- Newton / Quasi-Newton

Why bother with another (class of) algorithm?

Computing the gradient is costly ♡

For a given solution $\theta \in \mathbb{R}^p$ computing the gradient

$$\nabla F(\theta) = \mathbb{E}\Big[\frac{\partial f(\theta, \xi)}{\partial \theta}\Big]$$

is costly as it requires to compute a multidimensionnal integral (if ξ admits a density), or a large sum.

Indeed, in most machine learning application, we consider that ξ is uniformly distributed over the data (*empirical risk minimization*), thus computing the gradient require a pass over every sample in the dataset.

Dataset of size $N > 10^6$ are common.

Estimating the gradient

Instead of using a true gradient

$$g^{(k)} = \nabla F(x^{(k)})$$

we can use a *statistical estimator* of the gradient

$$\hat{g}^{(k)} \rightsquigarrow g^{(k)} = \mathbb{E}\Big[\frac{\partial f(\theta^{(k)}, \boldsymbol{\xi})}{\partial \theta}\Big]$$

The most standard estimator being

$$\hat{g}^{(k)} = \frac{\partial f(\theta^{(k)}, \xi^{(k)})}{\partial \theta}$$

where $\xi^{(k)}$ is drawn randomly according to the law of $\boldsymbol{\xi}$ (i.e. it is a random datapoint).

Wraping-up: GD versus SGD

Recall we want to solve the problem

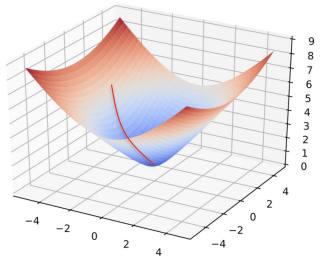
$$\min_{\theta \in \mathbb{R}^p} F(\theta) = \mathbb{E}\Big[f(\theta, \boldsymbol{\xi})\Big].$$

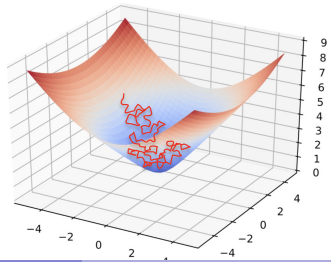
Gradient descent

$$\theta^{(k+1)} = \theta^{(k)} - \alpha\, \mathbb{E}\Big[\frac{\partial f(\theta^{(k)}, \boldsymbol{\xi})}{\partial \theta}\Big]$$

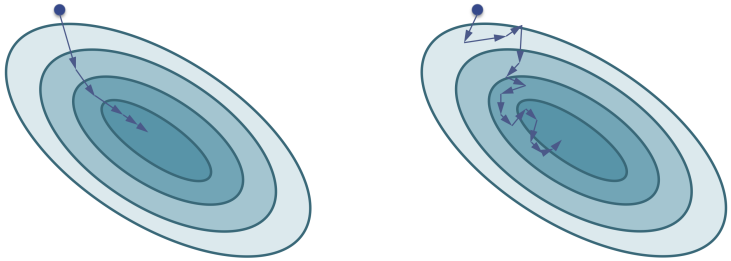
Stochastic gradient descent

$$\theta^{(k+1)} = \theta^{(k)} - \alpha\, \frac{\partial f(\theta^{(k)}, \xi^{(k)})}{\partial \theta}$$





A noisy trajectory



Gradient Descent

Stochastic Gradient Descent

Pros and Cons

Pros:

- computing $\hat{g}^{(k)} = \frac{\partial f(\theta^{(k)}, \xi^{(k)})}{\partial \theta}$ is really easy
- we do not need to spend lots of time early to get a precise gradient
- we can stop whenever we want (do not need a full pass on the data)
- Noise can help escape poor local minima or saddle points.

Cons:

- $\hat{g}^{(k)}$ is a noisy estimator of the gradient
- requires a new convergence theory
- $\theta^{(k+1)} := \theta^{(k+1)} - \alpha \hat{g}^{(k)}$ generally does not converges almost surely to the optimal solution as this make a noisy trajectory

Convergence intuition for SGD ♡

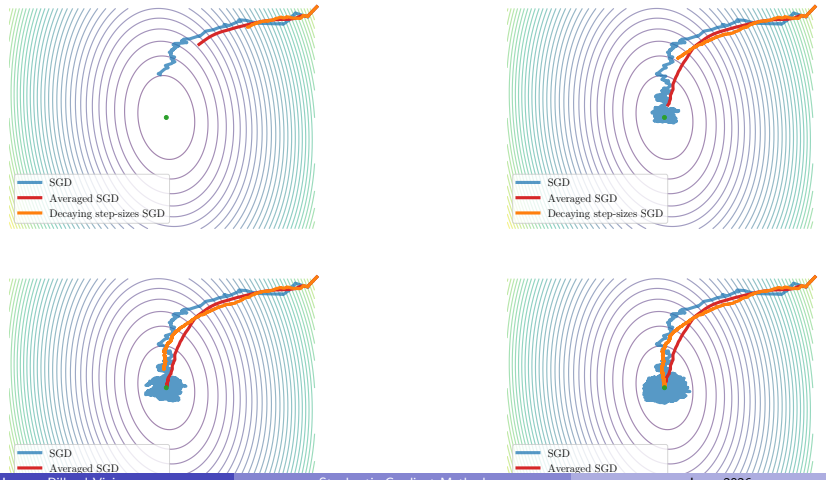
We consider the stochastic update

$$\theta^{(k+1)} = \theta^{(k)} - \alpha^{(k)} \hat{g}^{(k)}, \qquad \mathbb{E}[\hat{g}^{(k)} \mid \theta^{(k)}] = \nabla F(\theta^{(k)}).$$

- **Constant step size** ($\alpha^{(k)} = \alpha$): fast initial progress, but the noise does not vanish.
  $\Rightarrow$ under standard assumptions, the iterates typically keep fluctuating and performance stabilizes in a neighborhood of the optimum (accuracy $\sim O(\alpha)$).
- **Decreasing step size** ($\alpha^{(k)} \searrow 0$, e.g. $\alpha^{(k)} \sim 1/k$): the noise is gradually damped.
  $\Rightarrow$ under standard assumptions, convergence to the optimum is possible, but the progress becomes slower.
- **Iterate averaging** (Polyak–Ruppert): average the iterates $\bar{\theta}^{(k)} = \frac{1}{k} \sum_{t=1}^{k} \theta^{(t)}$.
  $\Rightarrow$ improves stability and, in classical strongly convex settings, can achieve optimal asymptotic rates.

Trade-off: speed vs accuracy vs stability.

Noisy trajectory ♢



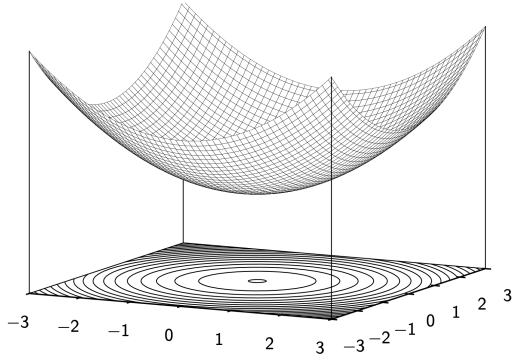
Mini-batch version ♢

- $\hat{g}^{(k)} = \frac{\partial f(x^{(k)}, \xi^{(k)})}{\partial \theta}$ is easy to compute but noisy estimator of the gradient
- $\hat{g}^{(k)} = \frac{1}{n} \sum_{i \in [n]} \frac{\partial f(x^{(k)}, \xi_i)}{\partial \theta}$ is long to compute but perfect estimator
- minibatch aims at a middle ground : randomly draw a sample S of realizations of ξ, and use

$$\hat{g}^{(k)} = \frac{1}{|S|} \sum_{\xi \in S} \frac{\partial f(x^{(k)}, \xi)}{\partial \theta}$$

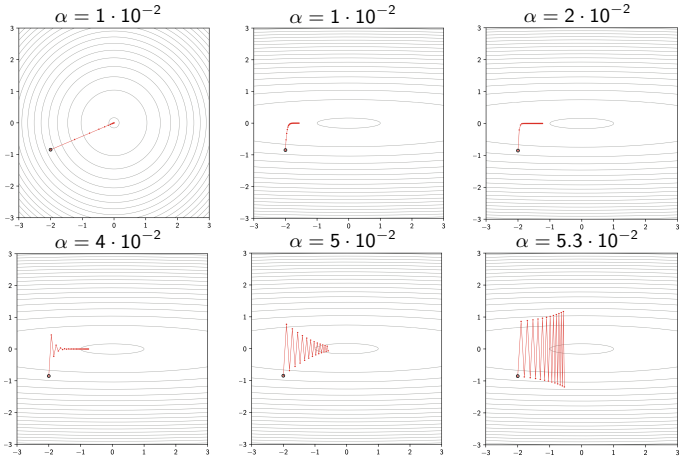
Accelerated methods ♢

The gradient descent method makes a strong assumption about the magnitude of the "local curvature" to fix the step size, and about its isotropy so that the same step size makes sense in all directions.



Accelerated methods

The gradient descent method makes a strong assumption about the magnitude of the "local curvature" to fix the step size, and about its isotropy so that the same step size makes sense in all directions.



Momentum methods

- Some optimization methods leverage higher-order information, in particular second order (Hessians) to use a more accurate local model of the functional to optimize. **They are too costly to use in practice.**
- However for a fixed computational budget, the complexity of these methods reduces the total number of iterations, and the eventual optimization is worse.
- Deep-learning generally relies on a smarter use of the gradient, using statistics over its past values to make a "smarter step" with the current one. **They are called momentum methods.**

The first improvement is the use of a "momentum" to add inertia in the choice of the step direction

$$\eta^k = \gamma \eta^{k-1} + \alpha \widehat{g}^k$$
$$\theta^{(k+1)} = \theta^{(k)} - \eta^k \ ,$$

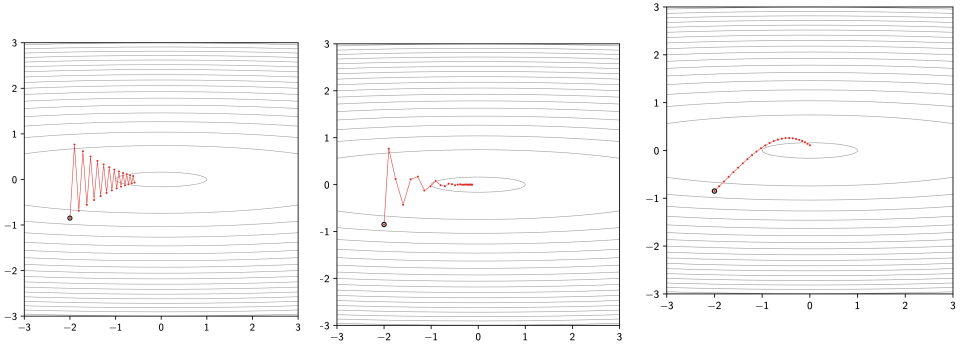
where $\widehat{g}^k$ is an estimator of the gradient at $\theta^{(k)}$ and $\gamma \in [0, 1]$ is a momentum parameter.
When $\gamma = 0$, this is the stochastic gradient descent method.
When $\gamma > 0$, this has nice properties:

- it is a first order method
- it has a memory of the past gradients and can go through local barriers
- accelerates if the gradient does not change too much : if no change new step size is $\alpha/(1-\gamma)$
- it dampens oscillations in narrow valleys

Accelerated methods

$\alpha = 5 \cdot 10^{-2},\ \gamma = 0$ $\qquad$ $\alpha = 5 \cdot 10^{-2},\ \gamma = 0.5$

Adam: momentum method using first and second moments to adapt the step size per direction.



What you have to know

When gradient descent is good but too costly, we can use stochastic gradient descent, which is a first order method that uses a noisy estimator of the gradient.

- Reduces (a lot!) the computational cost of the algorithm
- Allows to use large datasets
- Be carefull with the convergence theory, as the trajectory is noisy
- Can be used with a decreasing step size or averaging the points
- Can be used with a mini-batch version to reduce the noise
- Can be accelerated with inertia methods
- Is a modern trick you need to think about!

What you really should know

- GD vs SGD: full gradient vs stochastic estimates via cheaper computations; mini-batch is the middle ground.
- Step-size matters: constant vs decreasing schedules; iterate averaging stabilizes noisy trajectories.
- Momentum and adaptive methods: inertia to accelerate and damp oscillations; Adam uses first/second moments.
- Practical trade-offs: batch size controls variance vs cost.

What you have to be able to do

- Implement a basic SGD/minibatch loop and parameter update.
- Tune momentum and batch size; read loss curves to detect divergence/oscillation.
- Use decreasing step sizes or iterate averaging to improve stability/convergence.

What you should be able to do

- Explain convergence intuition under noise and how step-size schedules affect it.
- Design experiments comparing GD, SGD, and mini-batch on a dataset.
- Choose momentum/Adam settings and justify when each helps.
- Diagnose/mitigate issues: noisy plateaus, narrow valleys, poor local minima.