# Gradient algorithms

V. Leclère (ENPC)

May 2nd, 2025

# 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

# Contents

# Contents

# 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 $\varepsilon$-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

$$\underset{x \in \mathbb{R}^n}{\text{Min}} \qquad f(x)$$

- 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 $g(x) = \|x\|_1$;
  - ...

# Black-box optimization

We consider the following unconstrained optimization problem

$$\underset{x \in \mathbb{R}^n}{\text{Min}} \qquad f(x)$$

- 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 $g(x) = \|x\|_1$;
  - ...

# Contents

# 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 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 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 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:

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

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.

# 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:

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

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 $\tau_0$
    - automatically adapts to ensure convergence
    - more complex procedure exists

[3] There exists a lot of other alternatives

# Contents

# 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

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

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 $m > 0$

# 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

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

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 $m > 0$

# 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

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

# 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

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

# Strongly convex suboptimality certificate

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

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

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

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

# 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

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

# 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

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

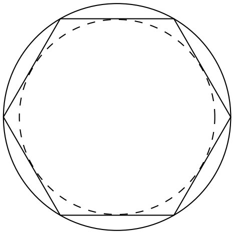
# Condition numbers

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

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



# 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

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

# 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

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

and thus

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

# 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

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

and thus

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

# Contents

# 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.

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

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

- Here $\|\cdot\|_{\maltese}$ could be any norm on $\mathbb{R}^n$.
  - If $\|\cdot\|_{\maltese} = \|\cdot\|_2$, the steepest descent is a gradient step, i.e. proportional to $-\nabla f(x^{(k)})$.
  - If $\|\cdot\|_{\maltese} = \|\cdot\|_P$ , $\|x\|_{\maltese} = \|P^{1/2} x\|_2$ 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 $\|\cdot\|_{\maltese} = \|\cdot\|_1$ , 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*

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

$\leadsto$ linear rate of convergence.

# Contents

# Solving a linear system

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



Alternatively, we can look to solve

$$\operatorname*{Min}_{x \in \mathbb{R}^n} \qquad f(x) := \frac{1}{2} x^\top A x - b^\top x$$

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 $A^\top A x = A^\top b$. However, this is not numerically stable, and should be avoided in practice.[6]

[6] Use GMRES or BiCGStab instead.

# Solving a linear system II

Alternatively, we can look to solve

$$\underset{x\in\mathbb{R}^n}{\text{Min}} \qquad f(x) := \frac{1}{2}x^\top A x - b^\top x$$

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 $A^\top A x = A^\top b$. 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.

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

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

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

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

# 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.

$$\left\langle u \, , v \right\rangle_A := u^\top A v = 0$$

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

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

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

# Conjugate direction method for quadratic function I ♢

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

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

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

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

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

# 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**

$$d^{(k)} = \tilde{d}^{(k)} - \sum_{i=1}^{k-1} \frac{\left\langle \tilde{d}^{(k)}, d^{(i)} \right\rangle_A}{\left\langle d^{(i)}, d^{(i)} \right\rangle_A} d^{(i)} \; ; \qquad \text{// A-orthogonalisation}$$

$$t^{(k)} = -\frac{\nabla f(x^{(k)})^\top d^{(k)}}{\left\langle d^{(k)}, d^{(k)} \right\rangle_A} \; ; \qquad \text{// optimal step}$$

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

**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

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

$$d^{(k)} = \tilde{d}^{(k)} - \sum_{i=1}^{k-1} \frac{-g^{(k)\top}(g^{(i+1)} - g^{(i)})}{d^{(i)\top}(g^{(i+1)} - g^{(i)})} d^{(i)}$$

$$= -g^{(k)} + \frac{g^{(k)\top}(g^{(k)} - g^{(k-1)})}{d^{(k-1)\top}(g^{(k)} - g^{(k-1)})} d^{(k-1)} = -g^{(k)} + \frac{\|g^{(k)}\|^2}{\|g^{(k-1)}\|^2} d^{(k-1)}$$

# Conjugate gradient algorithm - quadratic function



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

$$d^{(k)} = \tilde{d}^{(k)} - \sum_{i=1}^{k-1} \frac{-g^{(k)^\top} (g^{(i+1)} - g^{(i)})}{d^{(i)^\top} (g^{(i+1)} - g^{(i)})} d^{(i)}$$

$$= -g^{(k)} + \frac{g^{(k)^\top} (g^{(k)} - g^{(k-1)})}{d^{(k-1)^\top} (g^{(k)} - g^{(k-1)})} d^{(k-1)} = -g^{(k)} + \frac{\|g^{(k)}\|^2}{\|g^{(k-1)}\|^2} d^{(k-1)}$$

# Conjugate gradient algorithm - quadratic function

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

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

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

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



**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

  ;

  $t^{(k)} = \frac{\|g^{(k)}\|_2^2}{{d^{(k)}}^\top Ad^{(k)}}$ ; `// optimal step`

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

  $g^{(k+1)} = g^{(k)} + t^{(k)} Ad^{(k)}$ ; `// 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 $\beta$, 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.