# § Multidimensional optimisation (without constraints)

## I. Reminders... Math

1D ~ nD

**1D**

$f : \mathbb{R} \longrightarrow \mathbb{R}$

$f'$ —intuition→ slope of the tangent

$f''$

$\vdots$

**nD** — optimization / general

$f : \mathbb{R}^n \longrightarrow \mathbb{R}$ $\quad f : \mathbb{R}^n \longrightarrow \mathbb{R}^m$

⇝ Order 1 derivatives

$$f' = \lim_{h \to 0} \frac{f(\vec{x_0}+\vec{h}) - f(\vec{x_0})}{h}$$

**Taylor expansion formula (order n)**

$$f(x_0+h) = \underbrace{\sum_{k=0}^{n} \frac{f^{(k)}(x_0)}{k!} \times h^k}_{\text{polynomial } (h)} + o(h^k)$$

↓ order 1

$$f(x_0+h) = \underbrace{f(x_0) + f'(x_0) \cdot h}_{(*)} + o(h)$$

order 1 → linear fct → line → tangent

**Partial derivatives**

$$\frac{\partial f}{\partial x_i}(\vec{x^0})$$

↓

$f(x_1 \dots x_n)$ — all variables are fixed except $i$th one

$$x_i \longmapsto f(x_1^0 \dots x_{i-1}^0 \, x_i \, x_{i+1}^0 \dots x_m^0)$$

$\mathbb{R} \longrightarrow \mathbb{R}$

**Differential**

(*) $f$ differentiable :

$$f(\vec{x_0}+\vec{h}) = f(\vec{x_0}) + \underbrace{Df(x_0) \cdot \vec{h}}_{\text{linear fct}}$$

differential fct ↓ linear fct ...

$Df(x_0)$ linear ... ↓ matrix

$Df(\vec{x_0}) : \mathbb{R}^m \longrightarrow \mathbb{R}$

basis w/ m vects $\{e_1 \dots e_m\}$

↳ size of its matrix ?

$m \times m$ ( m cols ) 1 line

$Df(e_1) \; \downarrow \; \downarrow \; \downarrow \; Df(e_m) \in \mathbb{R}$

$Df(x_0) \longleftrightarrow$ $1 \times m$ matrix

canonical basis :

$$\left( \frac{\partial f}{\partial x_1} \quad \cdots \quad \frac{\partial f}{\partial x_m} \right)$$

$$\nabla f(x_0) = \begin{pmatrix} \frac{\partial f}{\partial x_1} \\ \vdots \\ \frac{\partial f}{\partial x_m} \end{pmatrix}$$

matrix of $Df \downarrow \nabla f^t$

1D $f' \longleftrightarrow$ nD $\nabla f$

**Taylor formula (order 1)**

$$f(\vec{x}_0 + \vec{h}) = f(\vec{x}_0) + \underbrace{Df(\vec{x}_0)}_{\text{linear}} \cdot \vec{h} + o(\|h\|)$$

$$= \nabla f(\vec{x}_0)^t \times \vec{h}$$

$$= \langle \nabla f(\vec{x}_0), \vec{h} \rangle$$

**Reminders**

$\vec{u}, \vec{v}$ vectors

$\langle \vec{u}, \vec{v} \rangle$ — 2 notations — $\vec{u} \cdot \vec{v}$

$= \vec{u}^t \times \vec{v}$

$\{$

$(A \times B)$

$u^t$ $(\cdots\cdots)$ $\times$ $v$ $\begin{pmatrix} \cdot \\ \vdots \\ \cdot \end{pmatrix}$ $= (\cdot)$ $1 \times 1 \leftrightarrow \mathbb{R}$

**Intuition behind the gradient**

Taylor formula

$$f(\vec{x}_0 + h) \sim f(\vec{x}_0) + \langle \nabla f(\vec{x}_0), \vec{h} \rangle$$

$$\Longleftrightarrow \quad \underbrace{f(\vec{x}_0 + \vec{h}) - f(\vec{x}_0)}_{\text{if } > 0 \text{ then } f \nearrow} \sim \underbrace{\langle \nabla f(\vec{x}_0), \vec{h} \rangle}_{\text{vector}}$$

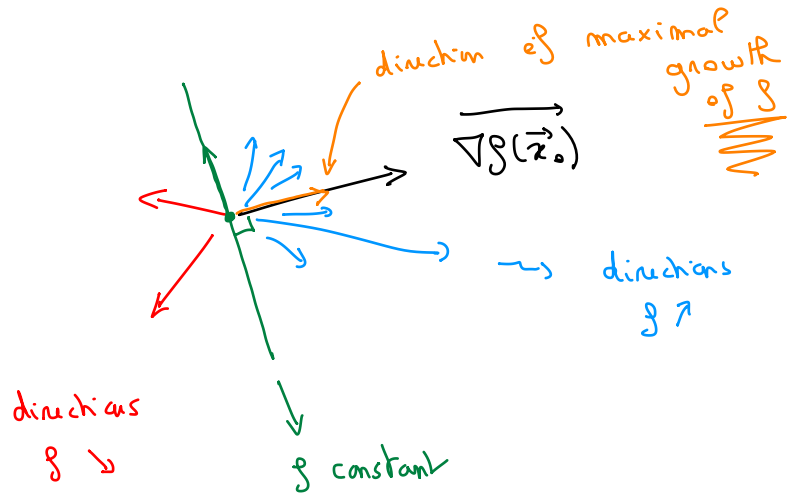
otherwise $f \searrow$

if $= 0$ then $f$ is constant

$\|\vec{h}\| = 1$

maximum for $h$ and $\nabla f(\vec{x}_0)$ aligned

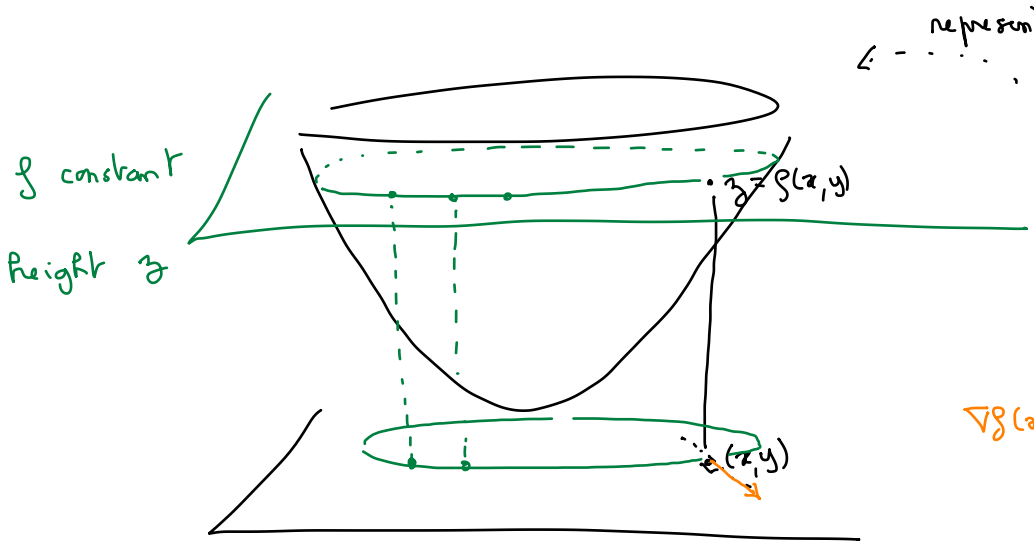
sign ?




$\vec{h}$ st $\langle \nabla f(x_0), \vec{h} \rangle > 0$

$\vec{h}$ st $\ldots < 0$

$\vec{h}$ st $\ldots = 0$



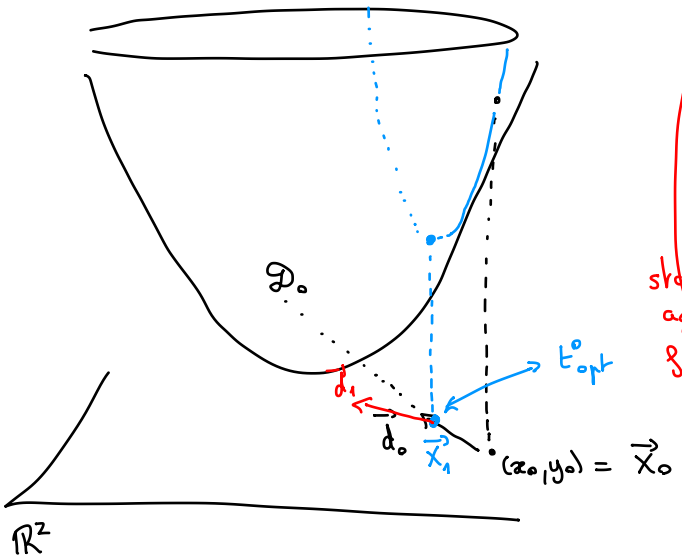

$f : \mathbb{R}^2 \longrightarrow \mathbb{R}$

$\nabla f(x,y)$

min $f$ ?

## II. Descent algorithms

Example $f : \mathbb{R}^2 \longrightarrow \mathbb{R}$ ⟶ represented by a surface




$f$ is decreasing along $\vec{d}_0$

$\mathcal{D}_0$ : half-line { starting from $\vec{x}_0$ with direction $\vec{d}_0$

$x_0 + t \cdot \vec{d}_0 \qquad t \in \mathbb{R}^+$

↓

find the min of $f$ along this line ⟹ $x_1$

start again from $x_1$

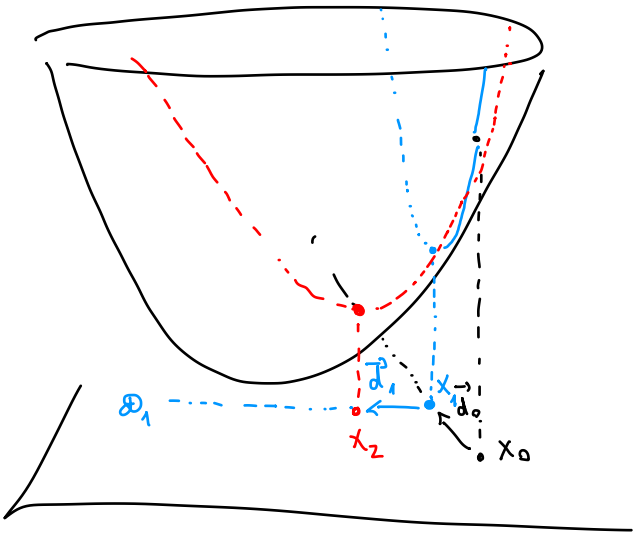
(*) 1D pb

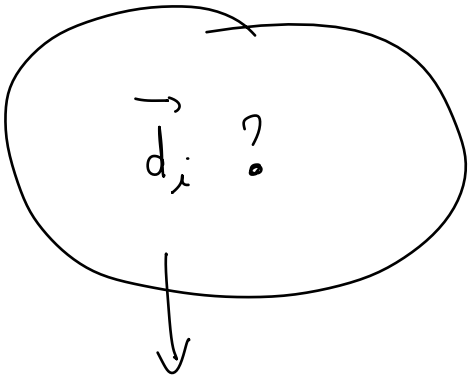
(*) find $t$ s.t $f(\vec{x}_0 + t \cdot \vec{d}_0)$ minimum

$t \longmapsto f(\vec{x}_0 + t \cdot \vec{d}_0)$ — min ?

$\mathbb{R} \longrightarrow \mathbb{R}$

1D optimization § 1




depends on the algorithm

Most basic

**gradient descent algorithm**

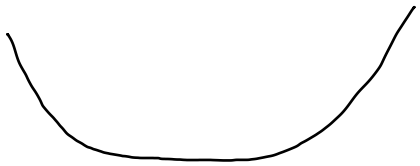
$$\vec{d}_i = - \nabla f(\vec{x}_i)$$

↳ greedy algorithm ....

⊖ not the fastest — many iterations ....

⊕ numerically very stable

order 1 methods



flat functions — challenging for opt. algorithms ...

**conjugate gradient descent algorithm**

⊕ faster

⊖ may fail with non convex functions ...

Order 2 methods — based on Newton-Raphson (nD)

↓

**Newton-Raphson method**

+

**Quasi-Newton methods**

approximations of Hessian

$$X_{i+1} = X_i - H(f)(X_i)^{-1} \times \nabla f(X_i)$$

nD Newton

1D

$f : \mathbb{R} \to \mathbb{R}$

↓ golden + parabolic

↘ Newton on $f'$

$f'(x) = 0$

$$x_{i+1} = x_i - \frac{f'(x_i)}{f''(x_i)}$$

intuition

nD

$\mathbb{R}^n$

$$X_{i+1} = X_i - \frac{\nabla f(X_i)}{\text{second order derivative of } f}$$

↓

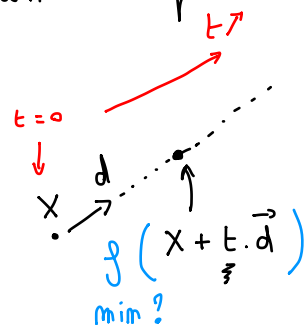
matrix — Hessian matrix

$H(f)(X_i)$

$\left( \frac{\partial^2 f}{\partial x_i \partial x_j} \right)$

## ① Gradient descent alg.

```
gradient_descent( f , ∇f , X0 , ε , δ , N )
    X ← X0          // X : current point
    iter ← 0
    X_prev ← X + ones(size(X))   // just to enter the while loop (init....)
    while ((||X - X_prev|| > ε) && (iter < N))
    {
        d ← - ∇f(X)
        // Optimize along the Ralf-line (X, d)
        φ ← @(t) f(X + t.d⃗)
        // 1D optim ( → Brent ( parabolic + golden ) → fminbnd
        here : 1Doptim (φ, [ ... ]   interval s.t. the fun is unimodal
        // Sweeping method to init the
        //          interval
```



$t_0 \leftarrow 0$

$t_1 \leftarrow \delta$

$y_0 \leftarrow \varphi(t_0)$

$y_1 \leftarrow \varphi(t_1)$

while $(y_0 \geq y_1)$ // move right

  $t_0 \leftarrow t_1$

  $y_0 \leftarrow y_1$

  $t_1 \leftarrow t_0 + \delta$

  $y_1 \leftarrow \varphi(t_1)$

end

// $t_1$ ens $t_{max}$ $t_1 - 2\delta$

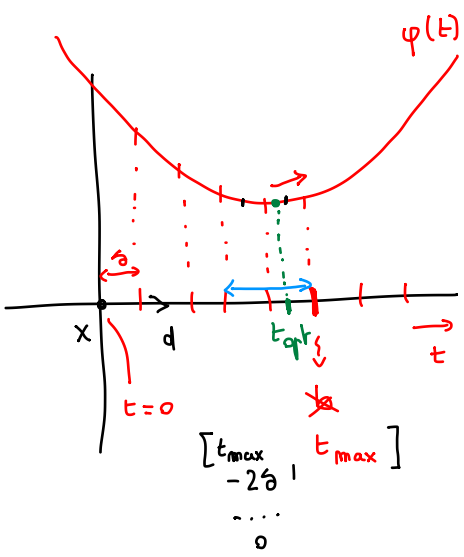
$t_{opt} \leftarrow$ 1Doptim $(\varphi, 0, t_1, \varepsilon, (N))$

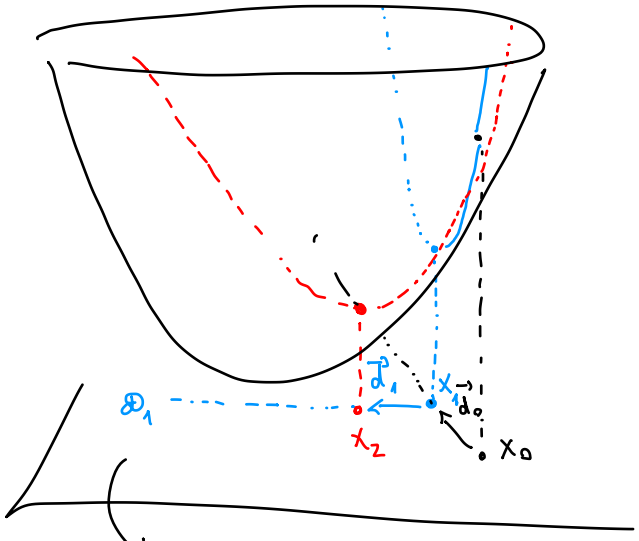
$x_{prev} \leftarrow x$

$x \leftarrow x + t_{opt} \cdot d$

end

end




stop : $\| x_i - x_{i+1} \| < \varepsilon$

$\Rightarrow x_i$ Cauchy sequence

$\downarrow$

stop $\leftrightarrow \| x_i - \overbrace{x^*}^{sol} \| < \varepsilon$

$t_{opt}$ — optimal for $\varphi$

$\uparrow$

Property optimal step gradient descent

$\forall i \quad d_i \perp d_{i+1}$

proof: $x_i \rightsquigarrow$ direction: $d_i = -\nabla g(x_i)$

$$\varphi(t) = f(x_i + t \cdot d_i)$$

$x_{i+1}$ .... $\min \to t_{opt}$

$$x_{i+1} = x_i + t_{opt} \cdot d_i \leftarrow$$

min of $\varphi$ $\Rightarrow$ $\varphi'(t_{opt}) = 0$

$$\varphi'(t) = \nabla f(x_i + t \cdot d_i) \times d_i$$

$\nabla f$: vector; $d_i$: vector; $\varphi'(t)$: $\mathbb{R}$; $\times$: $\cdot$

intuitively $\sim$ 1D

$f : \mathbb{R}^n \to \mathbb{R}$

$(f \circ g)' = f' \circ g \times g'$

$$\varphi'(t) = \nabla f(x_i + t \cdot d_i) \cdot d_i$$

At $t_{opt}$ $\Rightarrow$ $\varphi'(t_{opt}) = \nabla f(x_i + t_{opt} \cdot d_i) \cdot \vec{d}_i$

$\varphi'(t_{opt}) = 0$

$x_{i+1}$

$$\nabla f(x_{i+1}) = -\vec{d}_{i+1}$$

$\Leftrightarrow$

$$0 = d_{i+1} \cdot d_i$$

**Convergence**

If $f$ is $\mathcal{C}^2$ and elliptic then the algorithm converges

$\exists \alpha$ s.t. $\forall x, y$

$\| \nabla f(x) - \nabla f(y) \| \geq \alpha \| x - y \|$

"curvature"

**Reminder** – quadratic forms / polynomials of d°2 exactly

ex: $\mathbb{R}^3$ $\quad f(x,y,z) = x^2 - y^2 + 2z^2 + xz + 2yz$

can be represented by a matrix

$\Updownarrow$

$X^t \times A \times X$

$A = \begin{pmatrix} x & y & z \\ x & \cdot & \cdot & \cdot \\ y & & & \\ z & & & \end{pmatrix}$

symmetric

squared terms

$\cdot$ ⇝ $xz$ → splitted symmetrically

$$\begin{array}{c|ccc} & x & y & z \\ x & 1 & 0 & \frac{1}{2} \\ y & 0 & -1 & 1 \\ z & \frac{1}{2} & 1 & 2 \end{array}$$

$$f(x,y,z) = f(X) = X^t \cdot A \cdot X$$

$f(X) = A \cdot X$

$\nabla f(X) = A^t$

**Gradient of such a function:**

$$\nabla f = 2AX$$

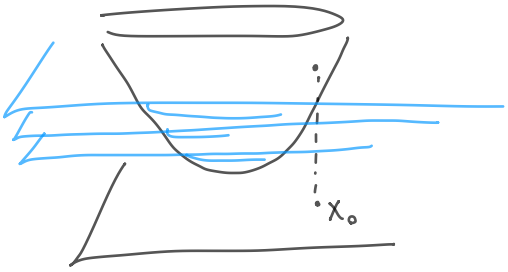
⚠ if $A$ symmetric

If $A$ not symmetric

$$\nabla f = (A + A^t) \cdot X$$

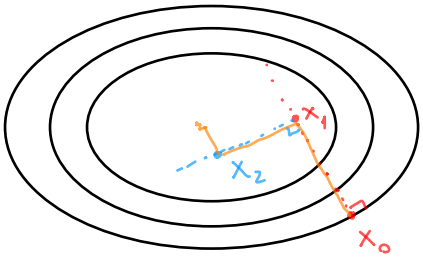
**Problem:** practical

$f_1(x,y) = \alpha x^2 + \beta y^2 \qquad \alpha, \beta > 0$




↳ steepest gradient descent converges to min

for this BASIC function




Expected: STOPs at the min

steps

The problem comes from:

→ $\vec{d}_i \perp \vec{d}_{i+1}$ ⇝ the choice is "limited"

⇝ always the same directions

⇝ if $\vec{d}_0$ does not cross the min no $\vec{d}_i$ will cross it ...

→ the alg. is greedy ...

**Conjugate gradient descent** improves the convergence

↳ designed for quadratic functions

formulae come from such functions — polynomial d° 2

↳ can be extended to $\mathcal{C}^2$ elliptic functions

## ② Conjugate gradient descent

Idea : $f$ is a quadratic function

$$f(X) = \underbrace{X^t \cdot A \cdot X}_{\text{order 2}} + \underbrace{{}^t B \cdot X}_{\text{order 1}} + \underbrace{\underset{\wedge}{\mathbb{R}}\, c}_{\text{order 0}}$$

order 1 → linear → matrix × X

order 0 → constant

$\{e_1 \dots e_n\}$ basis

$f : \mathbb{R}^n \longrightarrow \mathbb{R}$

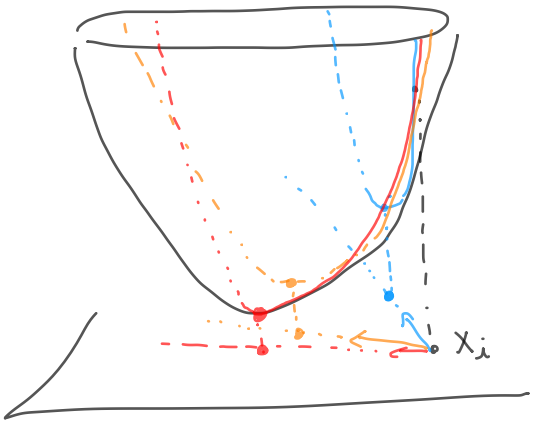
( matrix $\begin{pmatrix} \cdot & & \cdot \\ \downarrow & \cdots & \downarrow \end{pmatrix}$ n cols

$f(e_i) \in \mathbb{R}$

matrix size ? $1 \times n$

$\downarrow t$

$B$




each $\vec{d}$ ⟶ a min (the min of $f$ along line$(x_i, \vec{d})$)

which $\vec{d}$ gives the best minimum ?

$\vec{d} \nparallel -\nabla f(x_i)$

For quadratic function we can compute an exact formula for this "optimal" $\vec{d}$

This "optimal" $\vec{d}$ is a linear combination of $\nabla f(x_i)$ and previous direction $\vec{d}_{i-1}$

- Optimal direction:

$$\vec{d}_{i+1} = \nabla f(x_{i+1}) + \frac{\| \nabla f(x_{i+1}) \|^2}{\| \nabla f(x_i) \|^2} \vec{d}_i$$

Gradient Steepest descend

$\vec{d}_i = + \nabla f(x_i)$

$f$ quadratic $f(x) = \frac{1}{2} x^t . A . x + B^t . x + c$

$\nabla f(x) = Ax + B$

$\varphi(t) = f(x_i - t . \vec{d}_i)$

APg:

$x_0$ : close to the minimum ....

$x \leftarrow x_0$

$x_{prev} \leftarrow x + \begin{pmatrix} 1 \\ \vdots \\ 1 \end{pmatrix}$

$iter \leftarrow 0$

$d \leftarrow \nabla f(x)$ ⚠ $d$ : - descent direction

while $( (\| x - x_{prev} \| > \varepsilon) \;\&\&\; (iter < N) )$

{ $\varphi(t) = f(x - t . d)$

// min $\varphi$ ?

// if $f$ is quadratic

$$t_{opt} = \frac{\| \nabla f(x_i) \|^2}{x_i^t . A . x_i}$$

// if $f$ $\mathcal{C}^2$ elliptic

// 1D optim ⇝ find min $\varphi$

$t_0 = 0$

$t_1 = \delta$

$y_0 = \varphi(t_0)$

$y_1 = \varphi(t_1)$

while ( $y_0 \geq y_1$ )

$t_0 = t_1$

$y_0 = y_1$

$t_1 = t_0 + \delta$

$y_1 = f(t_1)$

end

$t_{opt} = 1D\text{optim}(\varphi, 0, t_1)$

$x_{prev} \leftarrow x$

$x \leftarrow x - t_{opt} \cdot d$

$d \leftarrow \nabla f(x) + \dfrac{\|\nabla f(x)\|^2}{\|\nabla f(x_{prev})\|^2} \cdot d$

end

**Convergence ...**

- case $f$ quadratic
- case $f$ $\mathcal{C}^2$ elliptic $\longrightarrow$ the alg. converges

When $f$ quadratic and $A$ definite positive : (all eigenvalues are $> 0$ $\Longleftrightarrow$ $^tX.A.X > 0$ $\forall X \neq 0$)

**Property**

$\forall i \quad \forall j < i \quad \vec{d}_i, \vec{d}_j$ are conjugate with respect to $A$

$\downarrow$

$\langle A d_j, d_j \rangle = 0$

conjugate $\leadsto$ orthogonal for the $A$-dot product

$\vec{d}_i$ conjugate with $\vec{d}_0 \ldots \vec{d}_{i-1}$

**Question** if $f : \mathbb{R}^n \longrightarrow \mathbb{R}$ quadratic elliptic ($\Longleftrightarrow$) / $A$ sym. definite positive

the algorithm ~~converges~~ steps in at most $n$ step.

For $\mathcal{C}^2$ functions $\Longrightarrow$ faster than gradient descent ....

Help ... $f(x) = \frac{1}{2} \cdot x^t \cdot A \cdot x + B^t \cdot x + c$

$\nabla f(x) = A \cdot x + B$

In the definition of $\vec{d}$ :

$\| \nabla f(x_i) \|^2 = \| A x_i + B \|^2 = \langle A x_i + B, A x_i + B \rangle$

$g(x) = x^t \cdot A \cdot x + B^t x + c$

$\nabla g(x) = 2 A x + B x$

Libraries $\rightsquigarrow$ many variants ....

pre-conditioned | conjugate gradients
bi-conjugate gradients .....

## III. Order 2 methods $\equiv$ Newton

$f : \mathbb{R}^n \longrightarrow \mathbb{R}$

Optimize $f$ | $x$ : min

$\Downarrow$ N.C.

$\nabla f(x) = 0$

$g = \nabla f$

### (1) Newton in $\mathbb{R}^n$ .... $\rightarrow$ Newton-Raphson

Goal $\rightarrow$ compute the | zeros / roots of $g$

$g : \mathbb{R}^n \longrightarrow \mathbb{R}^n$

$\nabla f : \mathbb{R}^n \longrightarrow \mathbb{R}^n$

$x \mapsto \nabla f(x)$

ex!

$$\mathbb{R}^3 \longrightarrow \mathbb{R}^3$$

$$g\begin{pmatrix} x \\ y \\ z \end{pmatrix} = \begin{pmatrix} x^2 + \cos(yz) \\ 2x + \ln(z) \\ xyz \end{pmatrix}$$

Newton $\longleftrightarrow$ solve non linear systems

$\Updownarrow$

$$g\begin{pmatrix} x \\ y \\ z \end{pmatrix} = \vec{0} \iff \begin{cases} x^2 + \cos(yz) = 0 \\ 2x + \ln(z) = 0 \\ xyz = 0 \end{cases}$$

Algorithm ....

Intuitively 1D $\rightsquigarrow$ nD

1D

$g(x) = 0$

Newton iteration:

$$x_{i+1} = x_i - \frac{g(x_i)}{g'(x_i)}$$

nD

$$\vec{x}_{i+1} = \vec{x}_i - \frac{g(\vec{x}_i)}{g'(x_i)} \quad ?$$

vector

$g : \mathbb{R}^n \longrightarrow \mathbb{R}^n$

~~$\nabla g$~~

$\frac{\partial g}{\partial x_i} \in \mathbb{R}^n$

Jacobian: $J(g)$

$f : \mathbb{R}^n \longrightarrow \mathbb{R}$

$$\nabla f = \begin{pmatrix} \frac{\partial f}{\partial x_1} \\ \vdots \\ \frac{\partial f}{\partial x_n} \end{pmatrix}$$

Jacobian — "set" of partial derivatives → matrix

$g : \mathbb{R}^n \longrightarrow \mathbb{R}^m$ / special case $f : \mathbb{R}^n \longrightarrow \mathbb{R}$

~~$g'$~~ — $Dg$ linear $\mathbb{R}^n \longrightarrow \mathbb{R}^m$

$\updownarrow$ matrix

$J(g)$ — Jacobian

Gradient $= \nabla f$

$g : \mathbb{R}^n \longrightarrow \mathbb{R}^n$

$g(\vec{x})$ vector dim $n$

$$= \begin{pmatrix} g_1(\vec{x}) \\ \vdots \\ g_n(\vec{x}) \end{pmatrix}$$

$g_i : \mathbb{R}^n \longrightarrow \mathbb{R}$

$$J(g) = \left( \frac{\partial g_i}{\partial x_j} \right) = \begin{pmatrix} — \nabla g_1^t — \\ \vdots \\ — \nabla g_n^t — \end{pmatrix}$$

$i, j \in \{1 \dots n\}$

ex:

$\mathbb{R}^3 \longrightarrow \mathbb{R}^3$

$$g\begin{pmatrix} x \\ y \\ z \end{pmatrix} = \begin{pmatrix} x^2 + \cos(yz) \\ 2x + \ln(z) \\ xyz \end{pmatrix} \begin{matrix} \leftarrow g_1 \\ \leftarrow g_2 \\ \leftarrow g_3 \end{matrix}$$

$g_i : \mathbb{R}^3 \longrightarrow \mathbb{R}$

$\nabla g_i$

Back to nD - Newton:

nD

$$\vec{x}_{i+1} = \vec{x}_i - \frac{g(\vec{x}_i)}{\cancel{g'(x_i)}} \quad ?$$

vector

$\cdots$ ? $\longleftrightarrow$ $J(g)(\vec{x}_i)^{-1} \times \cdots$

$J(g)(\vec{x}_i)$

matrix

## Newton - Raphson — $g: \mathbb{R}^n \to \mathbb{R}^n$ / $\boxed{g(x) = \vec{0}}$

$x_0$ : close to the solution ....

while ( $\| x_i - x_{i-1} \| > \varepsilon$ )

$$x_{i+1} = x_i - J(g)(x_i)^{-1} \times g(x_i)$$

end

$\sim \frac{g(x_i)}{g'(x_i)}$ | 1D ...

## Newton for optimization — $f: \mathbb{R}^n \to \mathbb{R}$ $\rightsquigarrow \min f$ ?

⇓ necessary condition

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

$g$ — Newton

$J(\nabla f)$ ?

$g = \nabla f$

$$\nabla f(x) = \begin{pmatrix} \frac{\partial f}{\partial x_1} \\ \vdots \\ \frac{\partial f}{\partial x_n} \end{pmatrix} \begin{matrix} \leftarrow g_1 \\ \vdots \\ \leftarrow g_n \end{matrix} \quad \rightsquigarrow g_i = \frac{\partial f}{\partial x_i}$$

$$J(\nabla f) = \begin{pmatrix} \frac{\partial g_i}{\partial x_j} \end{pmatrix} = \begin{pmatrix} \frac{\partial^2 f}{\partial x_i \partial x_j} \end{pmatrix}_{i,j \in \{1..n\}} = H(f)$$

derivate $f'$ (2)

Hessian matrix of $f$

## Newton for optimization — $f: \mathbb{R}^n \to \mathbb{R}$ $\rightsquigarrow \min f$ ?

⇓ necessary condition

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

$g$ — Newton

$x_0$ — close from the minimum

while ( $\| x_i - x_{i-1} \| > \varepsilon$ )

$$x_{i+1} = x_i - H(f)(x_i)^{-1} \times \nabla f(x_i)$$

end

inver a $n \times n$ matrix

Quasi-Newton methods ...

→ BFGS

→ .....

heuristics

approximate $H(\beta)^{-1}$ from $\nabla \beta$ ....