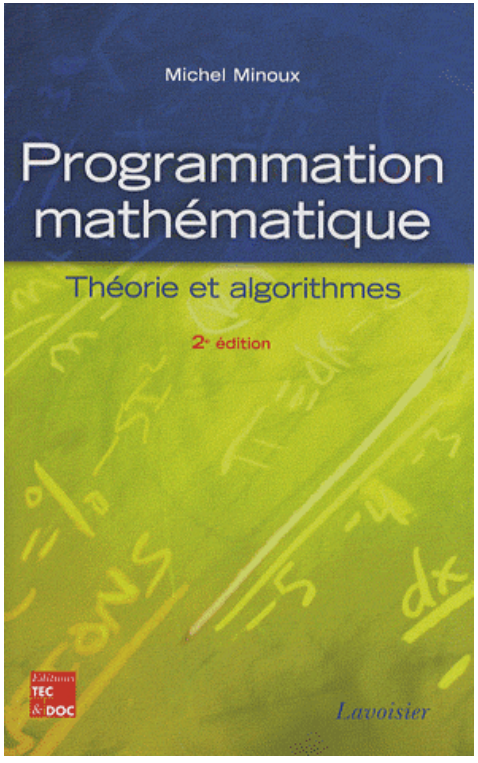
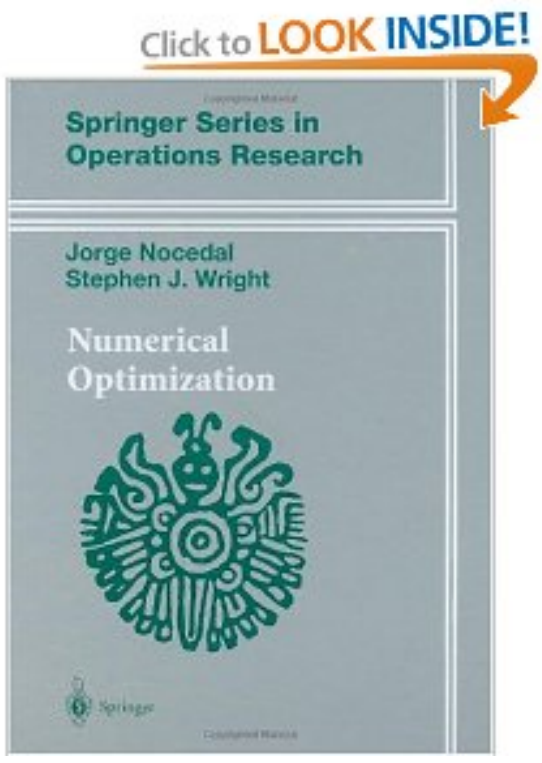
Metti 5

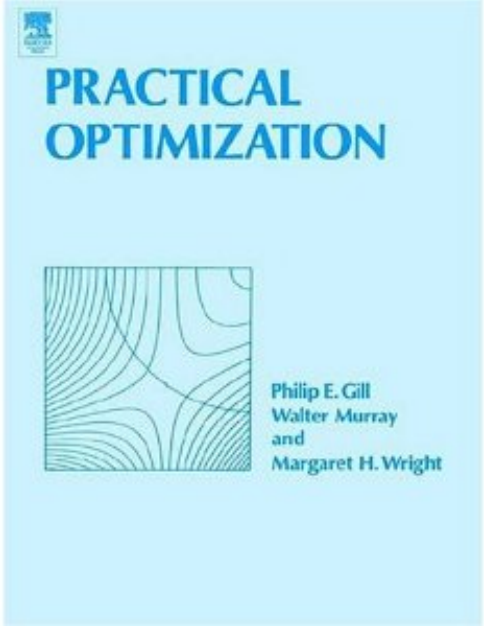
Optimization for nonlinear parameter estimation and function estimation

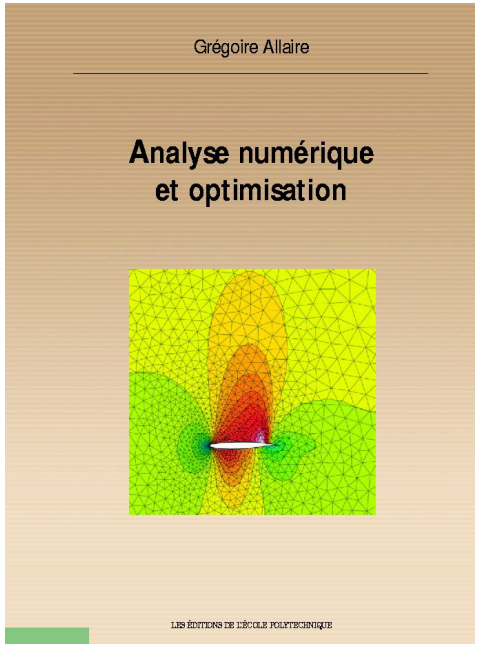
Lecture 7

– Roscoff, June 13-18, 2011 –


Michel Minoux
Programmation mathématique
Théorie et algorithmes
2e édition
TEC & DOC
Lavoisier


Click to LOOK INSIDE!
Springer Series in Operations Research
Jorge Nocedal
Stephen J. Wright
Numerical Optimization
Springer


PRACTICAL OPTIMIZATION
Philip E. Gill
Walter Murray
and
Margaret H. Wright


Grégoire Allaire
Analyse numérique et optimisation
LES ÉDITIONS DE L'ÉCOLE POLYTECHNIQUE

Objectives

Direct problem

$$\left.\begin{array}{r}\text{input}\\ \text{BC}\\ \text{IC}\\ \text{Parameters}\end{array}\right| \longrightarrow \text{Model} \longrightarrow \text{State solution}$$

Notations

$\mathcal{R}(u,\psi)=0$	Model
u	State
ψ	Unknown

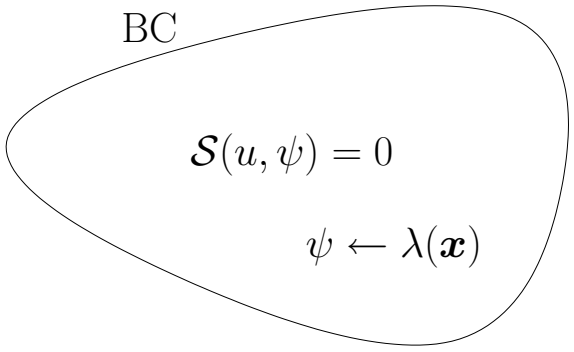
Inverse problem

From state measurements u_d, find unknown ψ that minimizes

$$j(\psi) := \mathcal{J}(u)$$

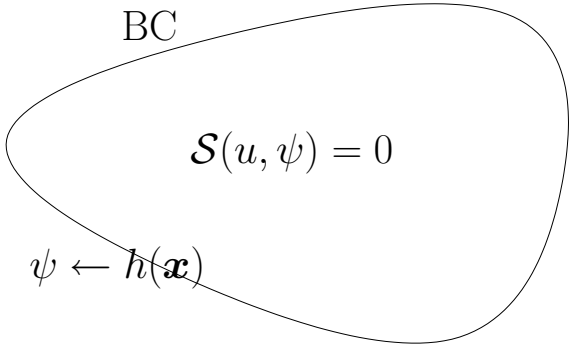
Examples

Thermal conductivity



- $\lambda = \mathsf{c}^{\mathsf{te}}$
- $\lambda(u \equiv T) \quad \Rightarrow \quad \lambda = \sum \lambda_i \xi^i(T)$
- $\lambda(\boldsymbol{x}) \quad \Rightarrow \quad \lambda = \sum \lambda_i \xi^i(\boldsymbol{x})$

Heat transfer coefficient



- $h = \mathsf{c}^{\mathsf{te}}$
- $h(u \equiv T)$
- $h(\boldsymbol{x})$

Inverse problem

From state measurements u_d, find unknown ψ that minimizes

$$j(\psi) := \mathcal{J}(u)$$

where

$$\mathcal{R}(u, \psi) = 0 \ : \ \psi \mapsto u$$

Contents

Non-linear optimization

Direct methods of the kind

of those seen in Lecture 2

usable

- for linear estimation,
- when $\dim \psi$ is "low"

We need Specific algorithms

- for non linear parameter estimation (iterations)
- for function estimation, i.e. $\dim \psi$ is "high"

Function $\longrightarrow$ parameters

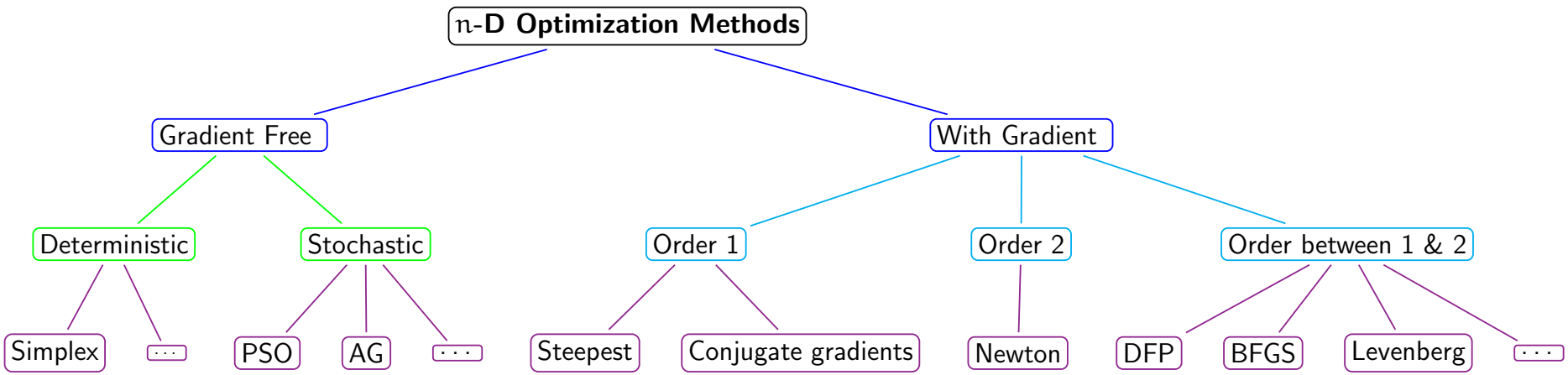
$$\psi \leftarrow \psi(\boldsymbol{s}) = \sum \psi_i \xi^i(\boldsymbol{s})$$

Optimization

We search

$$\bar{\psi} = \arg \min_{\psi \in \mathcal{K} \subset \mathcal{V}} j(\psi)$$

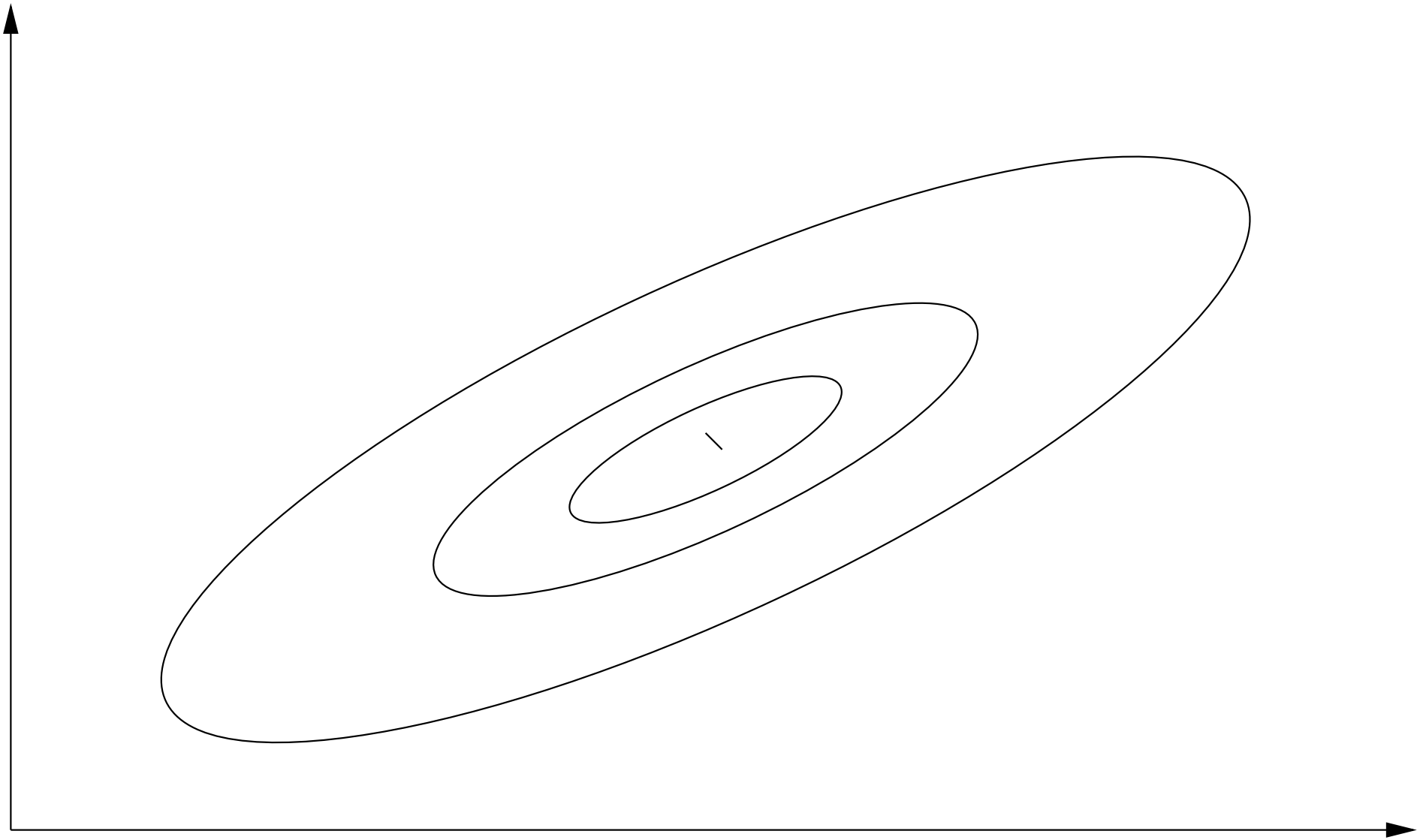
Methods (quite a lot . . .)



. . . and much more than that !

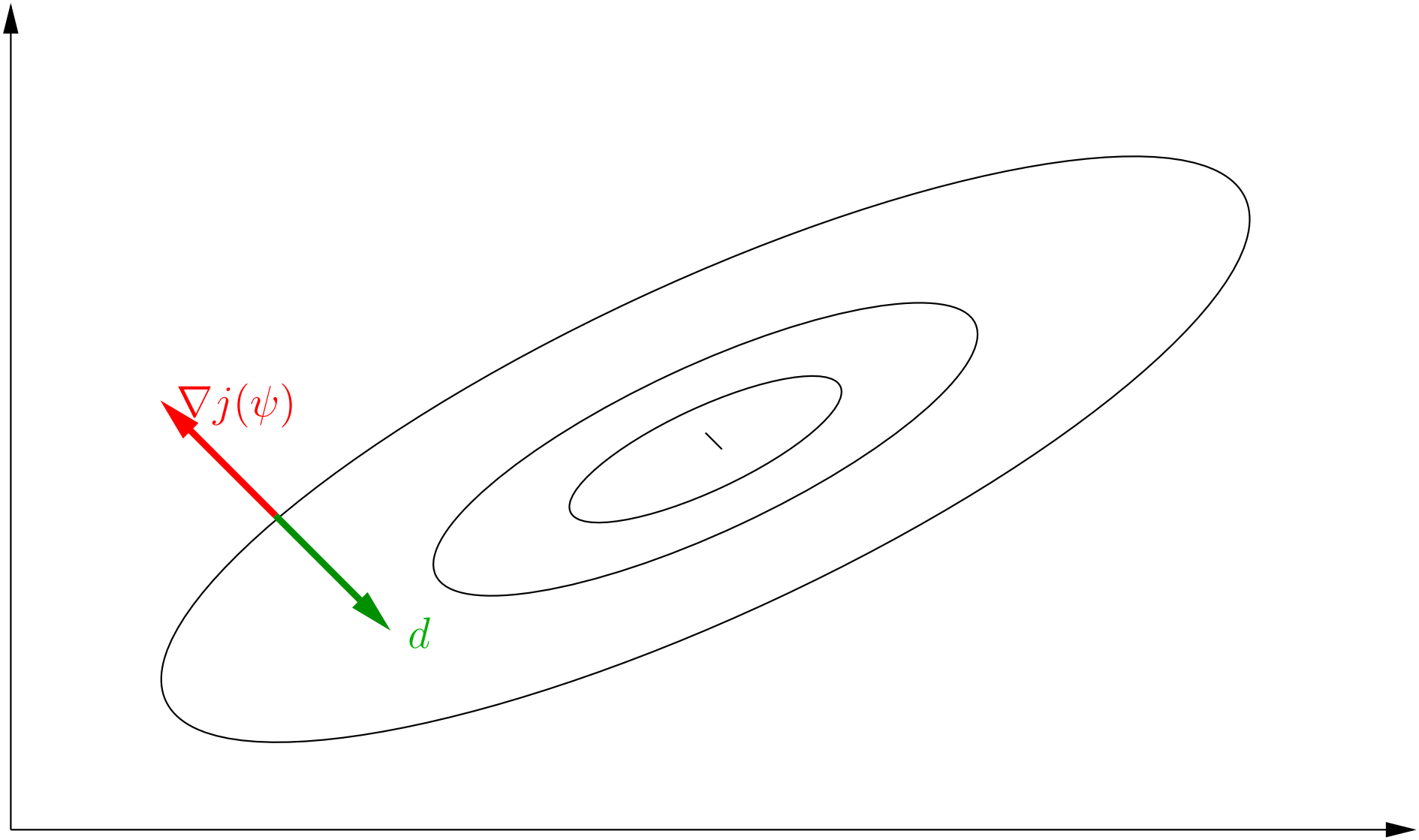
for gradient free, see [*OnWubolu,G.C. and Babu,B.V., New optimization techniques in engineering, Springer, 2003*]

Gradient-type methods



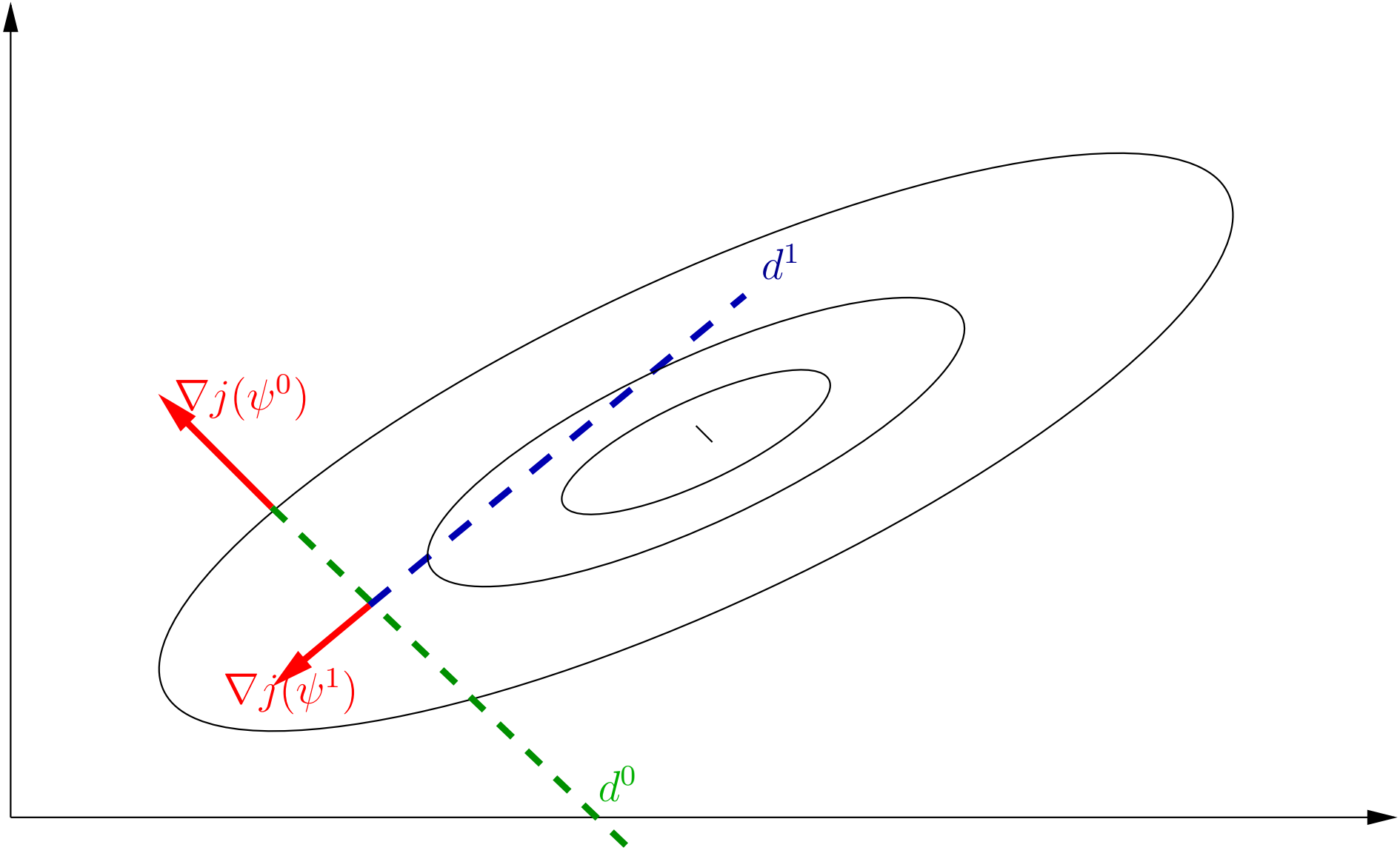
Gradient-type methods : Steepest method

First iteration



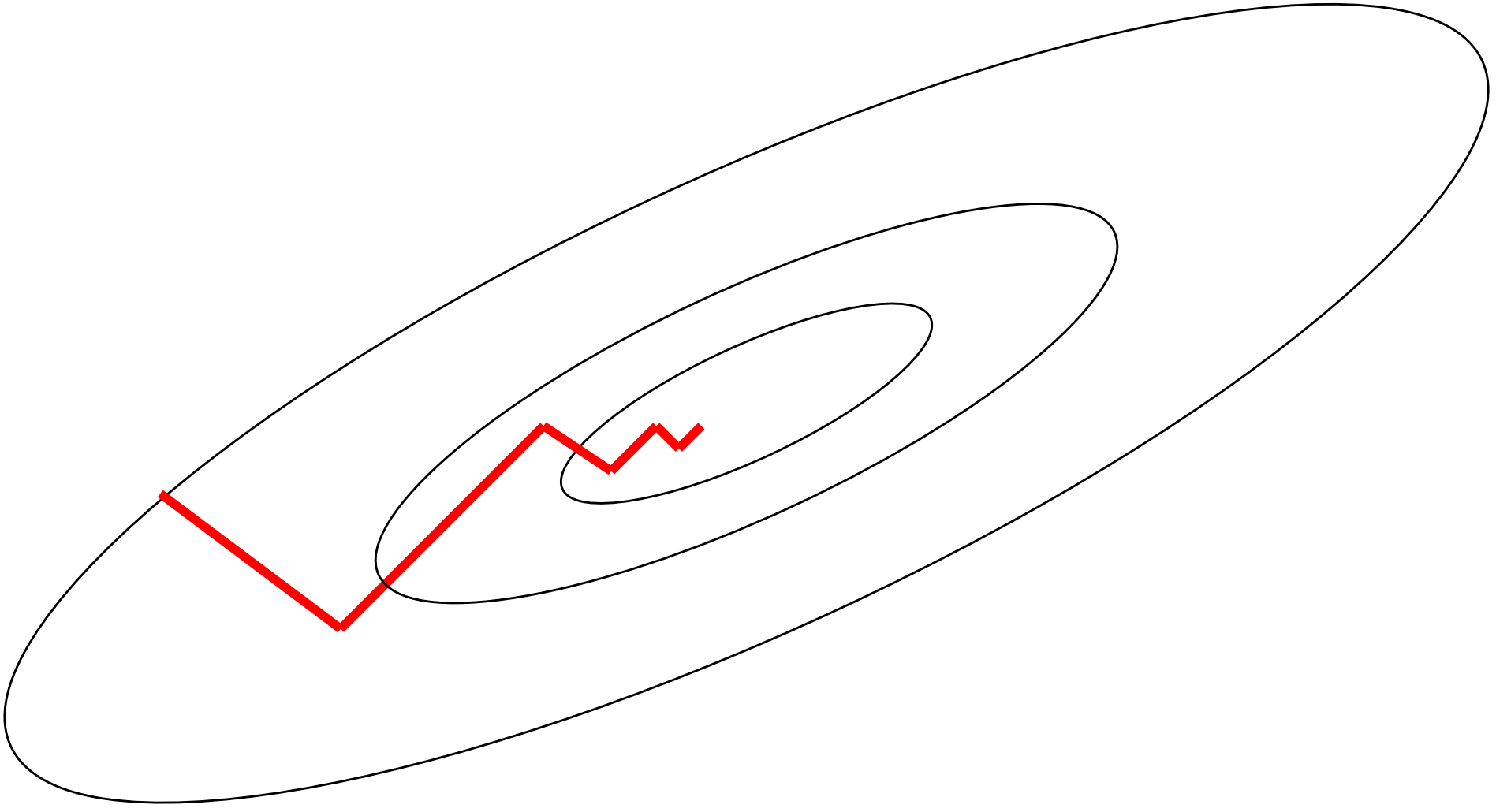
Gradient-type methods : Steepest method

Second iteration



Gradient-type methods : Steepest method

Successive displacement : Orthogonality $\longrightarrow$ zig-zag



Gradient-type methods : Steepest method

Algorithm 1: Steepest descent

while *(Stopping criterion not satisfied)* **do**

(We are at the point ψ^p, iteration p)

- compute the gradient $\nabla j(\psi)$
- the descent direction, $d^p = -\nabla j(\psi^p)$
- Line-search :

$$\text{Find } \bar{\alpha} = \arg\min_{\alpha>0} g(\alpha) = j\left(\psi^p + \alpha d^p\right)$$

Stopping criterion

$$\|\nabla j(\psi^p)\|_{2 \text{ or } \infty} \leq \varepsilon$$

$$\left|j(\psi^p) - j(\psi^{p-1})\right| \leq \varepsilon$$

$$\psi^p - \psi^{p-1} \leq \varepsilon$$

$$j(\psi^p) \leq \varepsilon$$

Gradient-type methods : Steepest method

Successive displacement : Orthogonality → zig-zag

Why such zig-zagging ?

Step p

Direction of descent :

$$d^p = -\nabla j(\psi^p)$$

Line search :

$$\text{Find } \bar{\alpha} = \arg\min_{\alpha>0} g(\alpha) = j\left(\psi^p + \alpha d^p\right)$$

So :

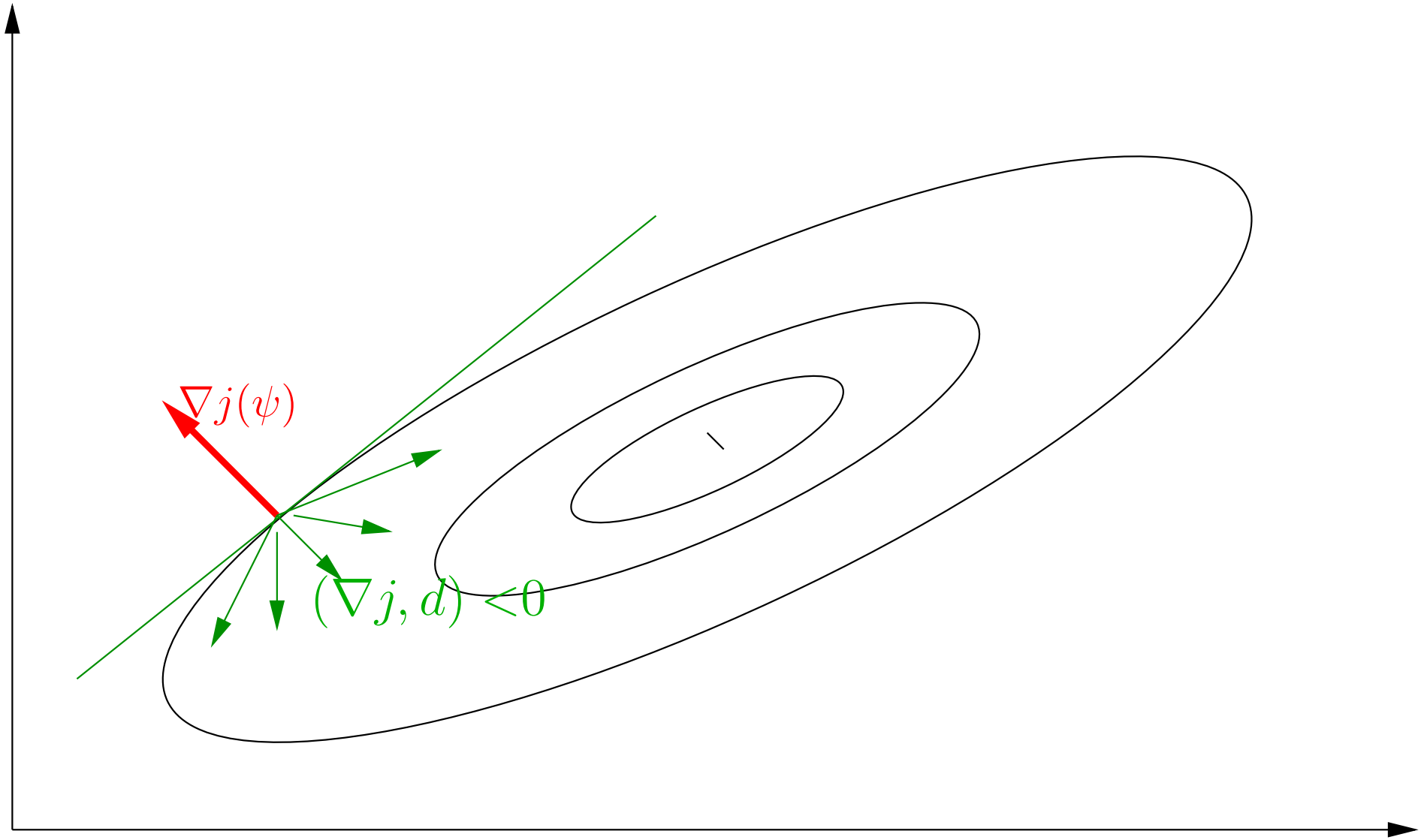
$$\begin{aligned} g'(\alpha^p) &= 0 \\ &= \left(d^p, \nabla j\left(\psi^p + \alpha d^p\right)\right) \\ &= \left(d^p, \nabla j\left(\psi^{p+1}\right)\right) \end{aligned}$$

So :

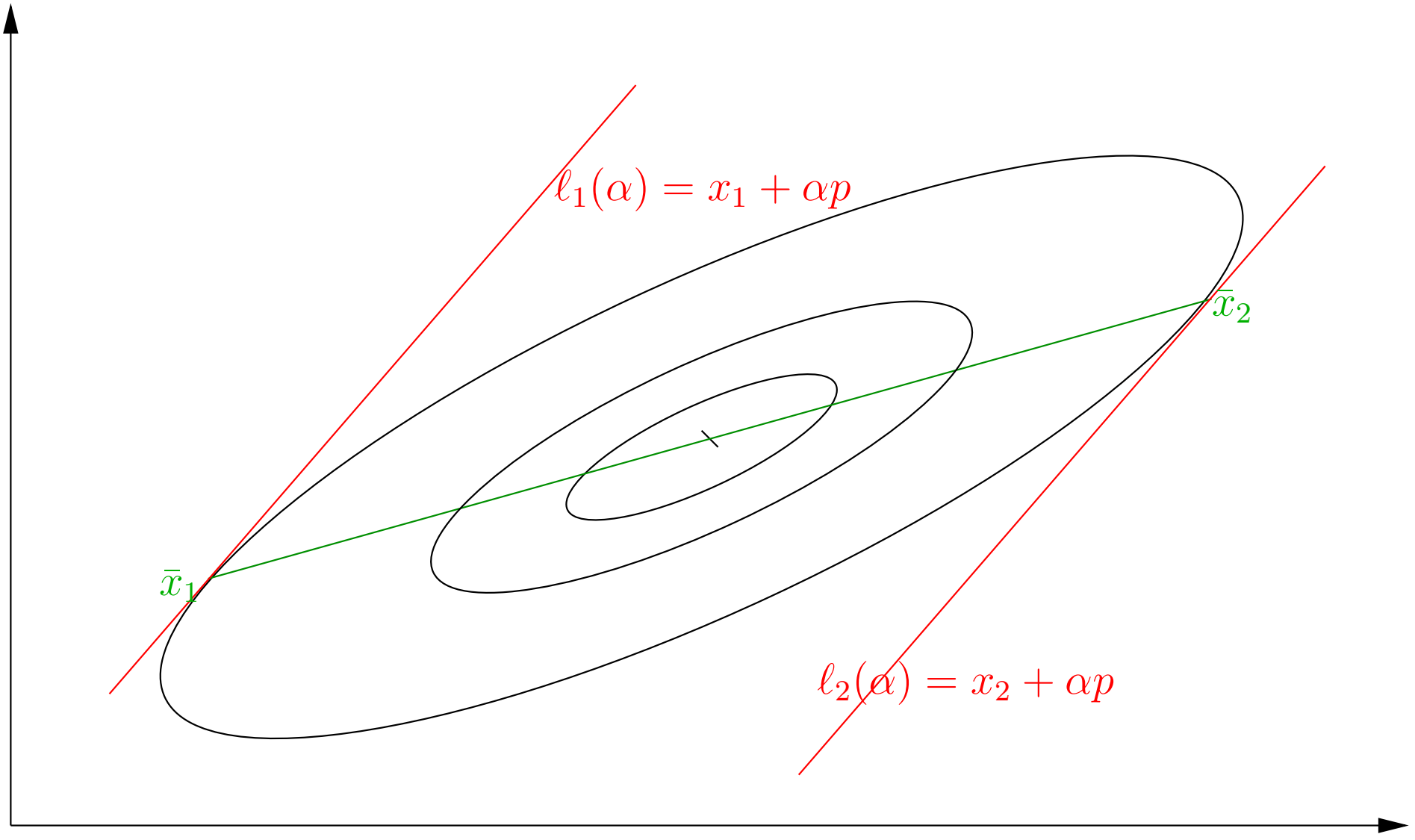
$$\left(d^p, d^{p+1}\right) = 0$$

Gradient-type methods

Admissible directions

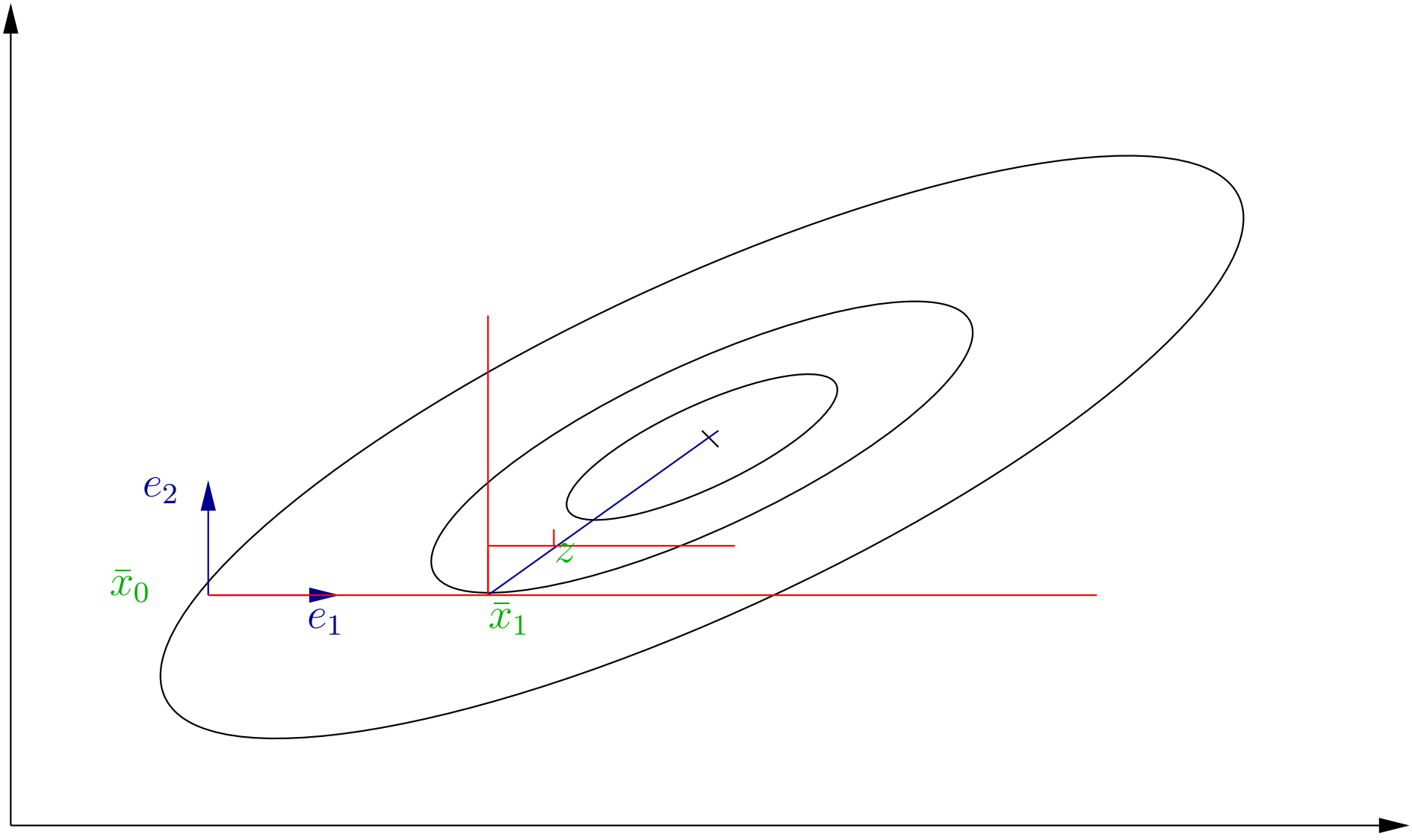


Conjugate directions



$\Rightarrow$ The vector $\bar{x}_1 - \bar{x}_2$ is conjugate to the direction p

Conjugate directions



$\Rightarrow$ The vector $z - \bar{x}_1$ is conjugate to the direction e_1

Conjugate directions for n–D

Algorithm

Let the quadratic cost

$$j(\psi) = \frac{1}{2}\left(\mathscr{A}\psi, \psi\right),$$

First iteration

$$d^0 = -\nabla j(\psi^0)$$

Then, from gradient orthogonality :

$$\begin{aligned}\left(d^0, \nabla j\left(\psi^1\right)\right) = 0 \quad &= \left(d^0, \mathscr{A}\psi^1\right) \\ &= \left(d^0, \mathscr{A}\left(\psi^0 + \alpha^0 d^0\right)\right) \\ &= \left(d^0, \mathscr{A}\psi^0\right) + \alpha^0\left(d^0, \mathscr{A}d^0\right).\end{aligned}$$

So we have the step length :

$$\alpha^0 = -\frac{\left(d^0, \mathscr{A}\psi^0\right)}{\left(d^0, \mathscr{A}d^0\right)}.$$

Step p

The direction d^p is chosen $\mathscr{A}-$conjugate to d^{p-1} :

$$\begin{aligned}\left(d^p, \mathscr{A} d^{p-1}\right) &= \left(-\nabla j\left(\psi^p\right) + \beta^p d^{p-1}, \mathscr{A} d^{p-1}\right) \\ &= -\left(\nabla j\left(\psi^p\right), \mathscr{A} d^{p-1}\right) + \beta^p \left(d^{p-1}, \mathscr{A} d^{p-1}\right) = 0\end{aligned}$$

So :

$$\beta^p = \frac{\left(\nabla j(\psi^p), \mathscr{A} d^{p-1}\right)}{\left(d^{p-1}, \mathscr{A} d^{p-1}\right)}.$$

Algorithm 2: The conjugate gradient algorithm applied on quadratic functions

Let $p = 0$, ψ^0 be the starting point,

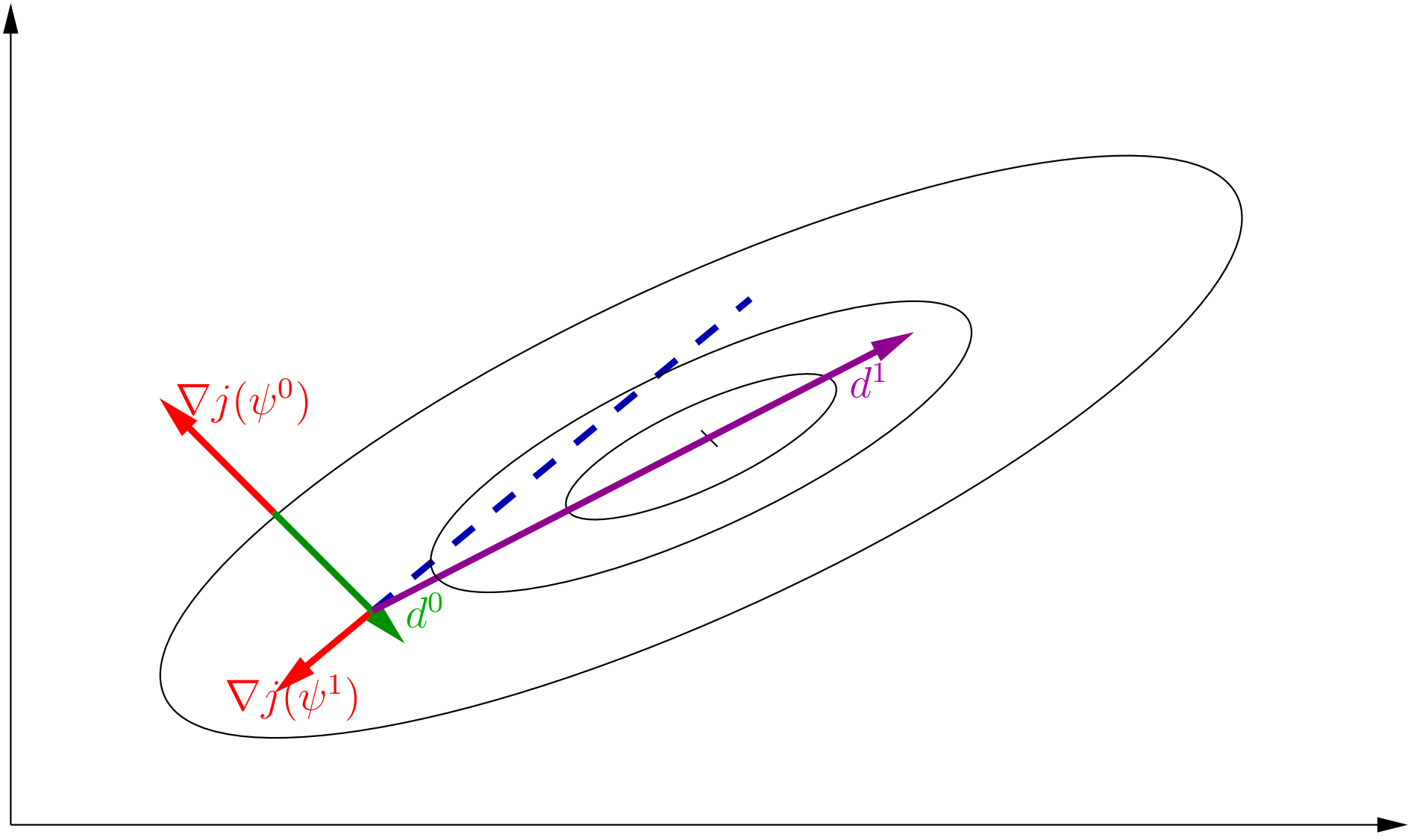
- Compute the gradient and the descent direction, $d^0 = -\nabla j(\psi^0)$,
- Compute the step size $\alpha^0 = -\dfrac{(d^0, \mathscr{A}\psi^0)}{(d^0, \mathscr{A}d^0)}$.

while *(Stopping criterion not satisfied)* **do**

At step p, we are at the point ψ^p.
We define $\psi^{p+1} = \psi^p + \alpha^p d^p$ with :

- the step size $\alpha^p = -\dfrac{(d^p, \nabla j(\psi^p))}{(d^p, \mathscr{A}d^p)}$
- the direction $d^p = -\nabla j(\psi^p) + \beta^p d^{p-1}$
- where the coefficient needed for conjugate directions : $\beta^p = \dfrac{(\nabla j(\psi^p), \mathscr{A}d^{p-1})}{(d^{p-1}, \mathscr{A}d^{p-1})}$;

Gradient-type methods



Conjugate gradients for non-quadratic functions

We use :

$$\begin{aligned}\nabla j(\psi^p) - \nabla j(\psi^{p-1}) &= \mathscr{A}\left(\psi^p - \psi^{p-1}\right)\\ &= \mathscr{A}\left(\psi^{p-1} + \alpha^{p-1} d^{p-1} - \psi^{p-1}\right)\\ &= \alpha^{p-1}\mathscr{A} d^{p-1},\end{aligned}$$

and combine with previously-seen relationships to get β^p through the

- Polak and Ribiere's method :

$$\beta^p = \frac{\left(\nabla j(\psi^p), \nabla j(\psi^p) - \nabla j(\psi^{p-1})\right)}{\left(\nabla j(\psi^{p-1}), \nabla j(\psi^{p-1})\right)},$$

- Fletcher and Reeves' method :

$$\beta^p = \frac{\left(\nabla j(\psi^p), \nabla j(\psi^p)\right)}{\left(\nabla j(\psi^{p-1}), \nabla j(\psi^{p-1})\right)}.$$

Conjugate gradients for non-quadratic functions

Algorithm 3: The conjugate gradient algorithm applied on arbitrary functions

Let $p = 0$, ψ^0 be the starting point,

$d^0 = -\nabla j(\psi^0)$,

perform the Line-search

while *(Stopping criterion not satisfied)* **do**

At step p, we are at the point ψ^p ; we define $\psi^{p+1} = \psi^p + \alpha^p d^p$ with :

- the step size $\alpha^p = \arg \min_{\alpha \in \mathbb{R}^+} g(\alpha) = j\left(\psi^p + \alpha d^p\right)$ with :
- the direction $d^p = -\nabla j(\psi^p) + \beta^p d^{p-1}$ where
- the conjugate condition parameter β^p satisfies either
  - Polak and Ribiere's method
  - Fletcher and Reeves' method

Newton

Assume that $j(\psi)$ is

- twice continuously differentiable,
- that second derivatives exist

Approach $j(\psi)$ by its quadratic approximation

$$\nabla j(\psi^{p+1}) = \nabla j(\psi^p) + \left[\nabla^2 j(\psi^p)\right] \delta\psi^p + \mathcal{O}\left(\delta\psi^p\right)^2,$$

so that

$$\left[\nabla^2 j\left(\psi^p\right)\right] \delta\psi^p = -\nabla j(\psi^p)$$

with

$$\psi^{p+1} = \delta\psi^p + \psi^p$$

Convergence rate

Quadratically

But

- difficult to compute $\nabla^2 j$, expensive
- convergence ensured only if $\nabla^2 j$ is positive definite

Quasi-Newton

Newton

$$\psi^{p+1} = \psi^p - \left[\nabla^2 j\left(\psi^p\right)\right]^{-1} \nabla j(\psi^p).$$

Idea :

$$\left[\nabla^2 j\left(\psi^p\right)\right]^{-1} \leftarrow \mathbf{H}^p$$

$$\mathbf{H}^{p+1} = \mathbf{H}^p + \Lambda^p$$

Imposed condition

$$\mathbf{H}\left[\nabla j(\psi^p) - \nabla j(\psi^{p-1})\right] = \psi^p - \psi^{p-1}$$

Different methods for the correction Λ^p

Quasi-Newton

Different methods for the correction Λ^p

- We set $\delta^p = \psi^{p+1} - \psi^p$ and $\gamma^p = \nabla j(\psi^{p+1}) - \nabla j(\psi^p)$

$\longrightarrow$ Davidon-Fletcher-Powell

$$\mathbf{H}^{p+1} = \mathbf{H}^p + \frac{\delta^p (\delta^p)^t}{(\delta^p)^t \gamma^p} - \frac{\mathbf{H}^p \gamma^p (\gamma^p)^t \mathbf{H}^p}{(\gamma^p)^t \mathbf{H} \gamma^p}$$

$\Rightarrow$ Broyden – Fletcher – Goldfarb – Shanno

$$\mathbf{H}^{p+1} = \mathbf{H}^p + \left[1 + \frac{\gamma^{p\,t} \mathbf{H}^p \gamma^p}{\delta^{p\,t} \gamma^p}\right] \frac{\delta^p (\delta^p)^t}{(\delta^p)^t \gamma^p} - \frac{\delta^p \gamma^{p\,t} \mathbf{H}^p + \mathbf{H}^p \gamma^p \delta^{p\,t}}{\delta^{p\,t} \gamma^p}.$$

Convergence rate

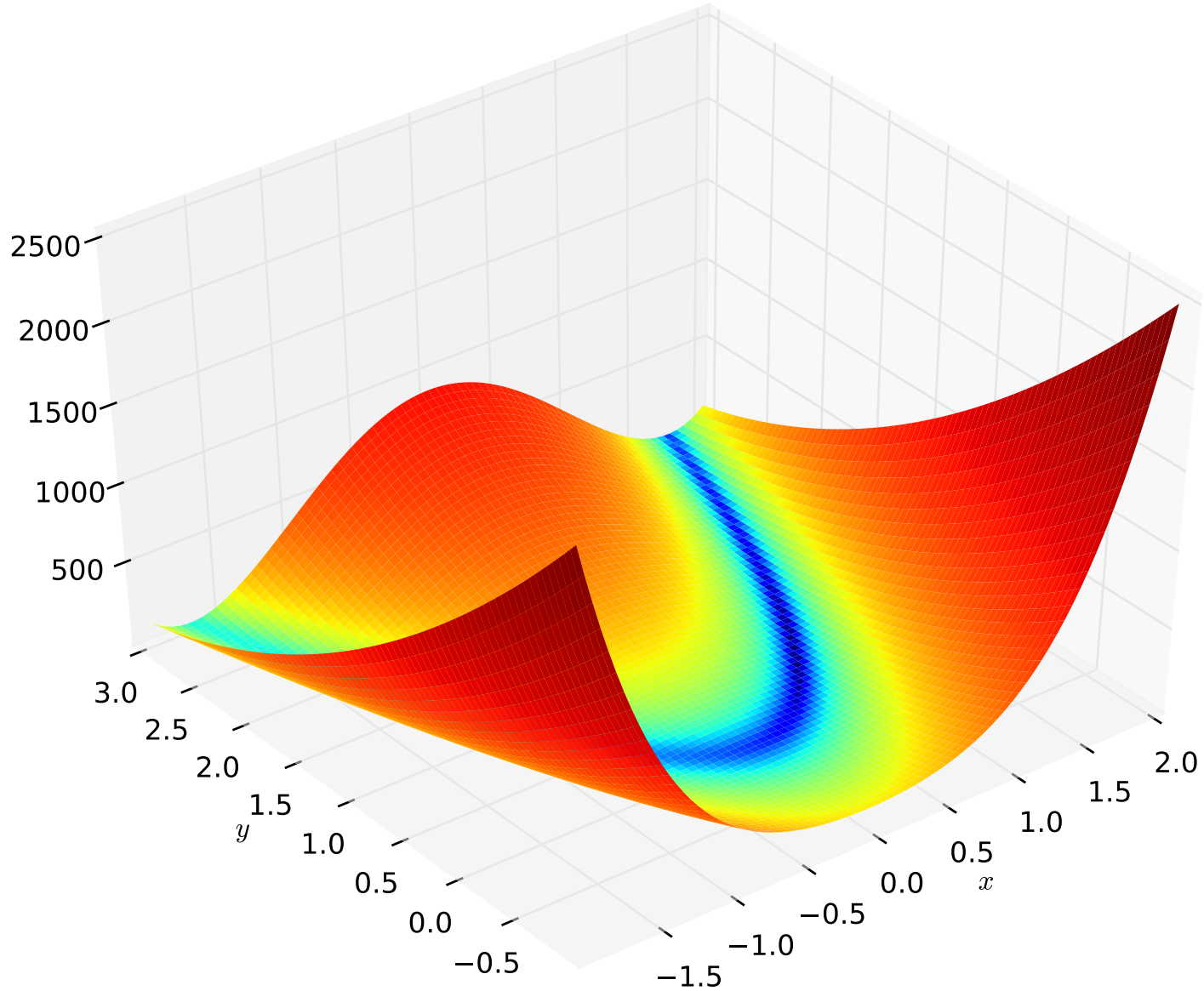
Superlinear

Remark

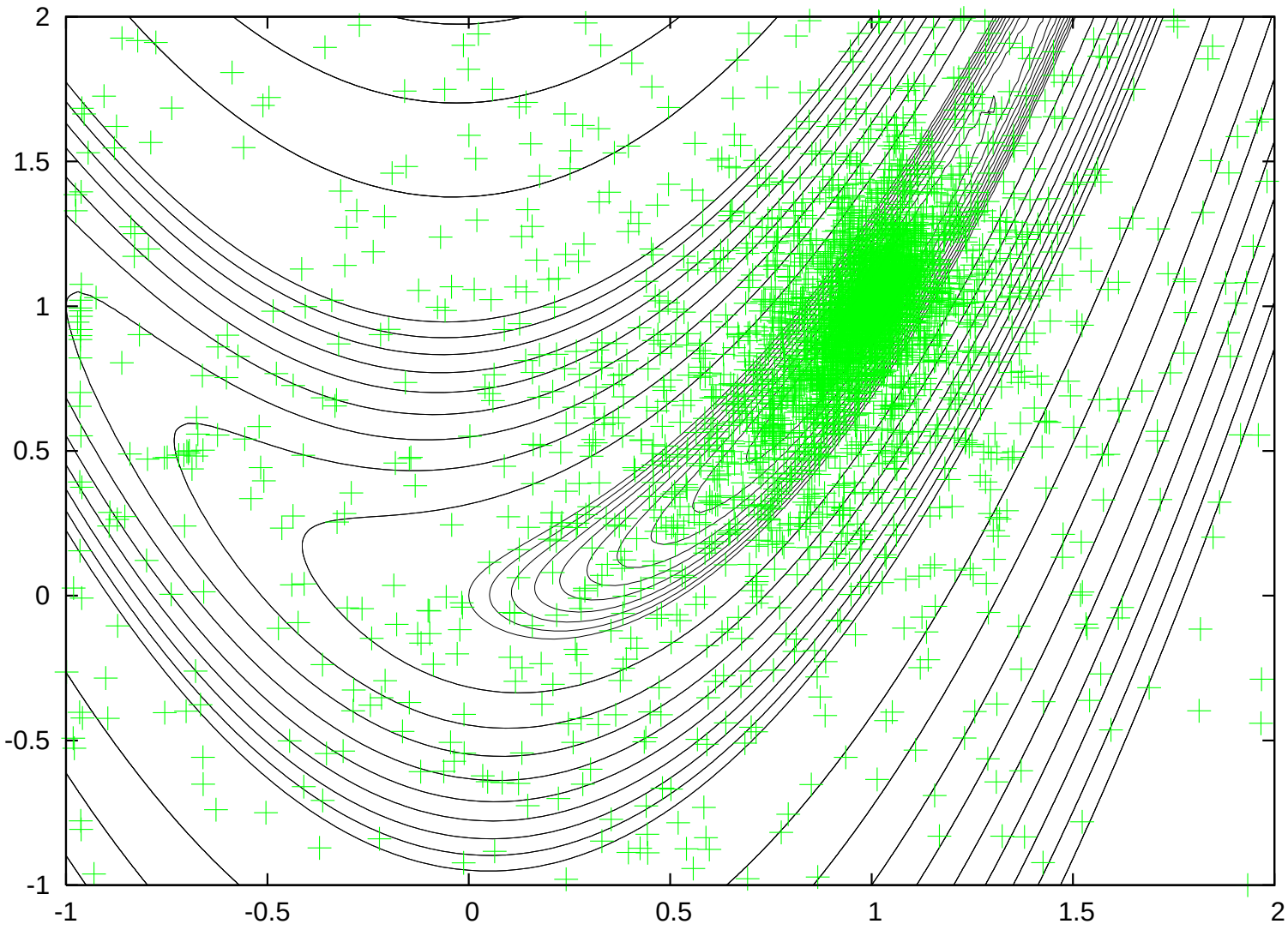
BFGS is less sensitive than DFP to line-search inacuracy

Test : Rosenbrock

Guess : $\begin{pmatrix} -1 \\ 1 \end{pmatrix}$, Optimum : $\begin{pmatrix} 1 \\ 1 \end{pmatrix}$

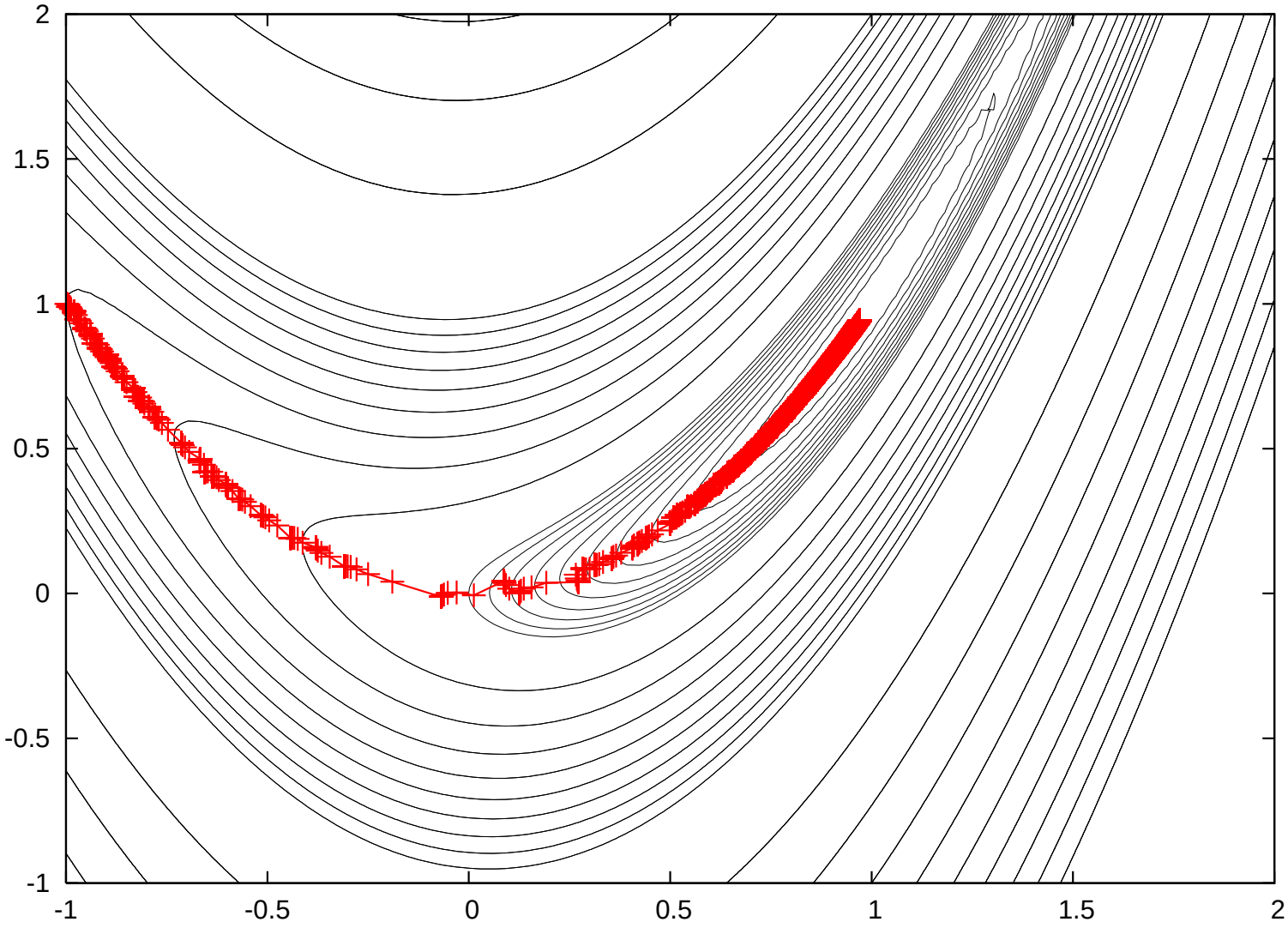


PSO



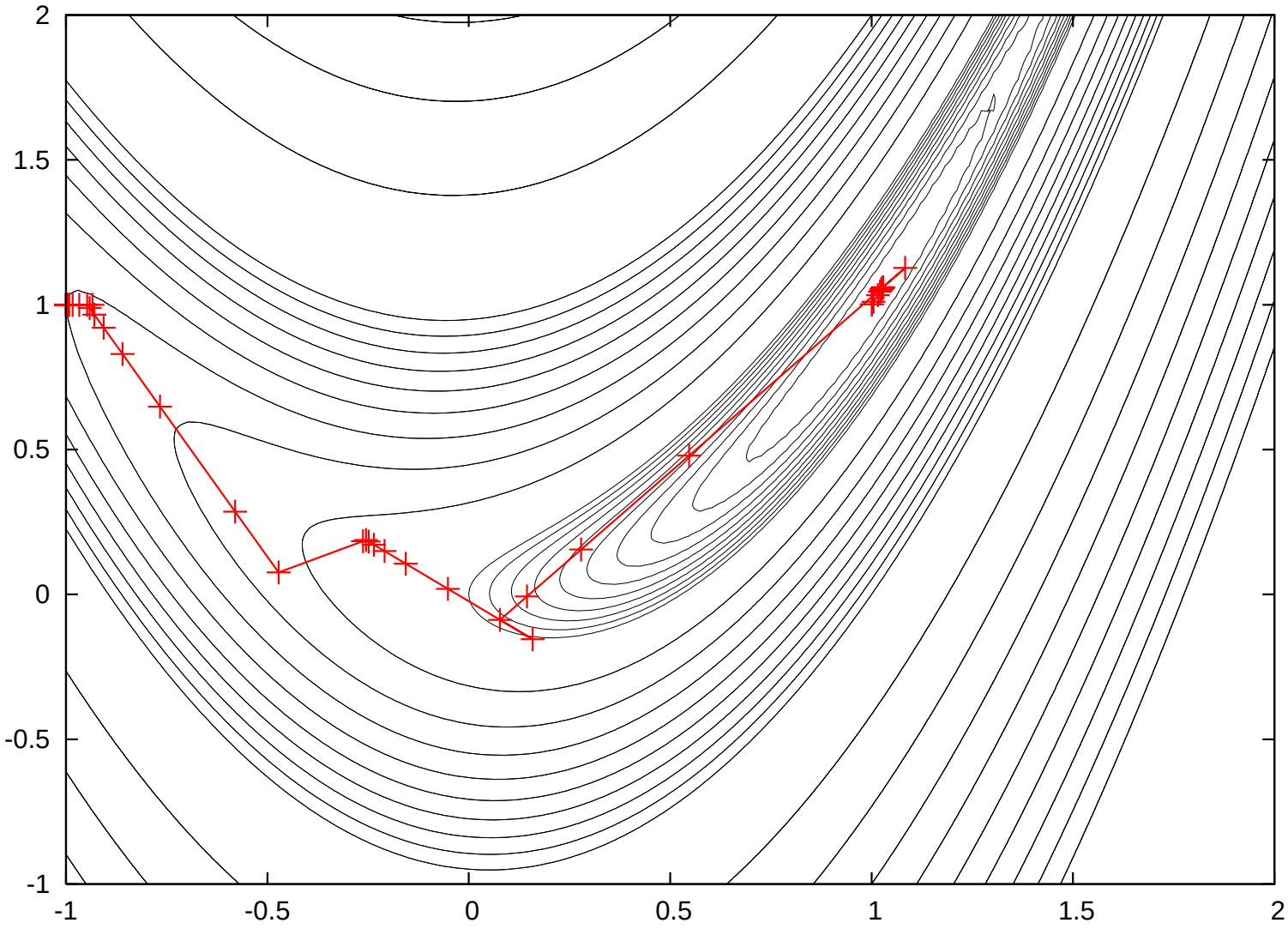
▷▷ http ://clerc.maurice.free.fr/pso/

Steepest descent



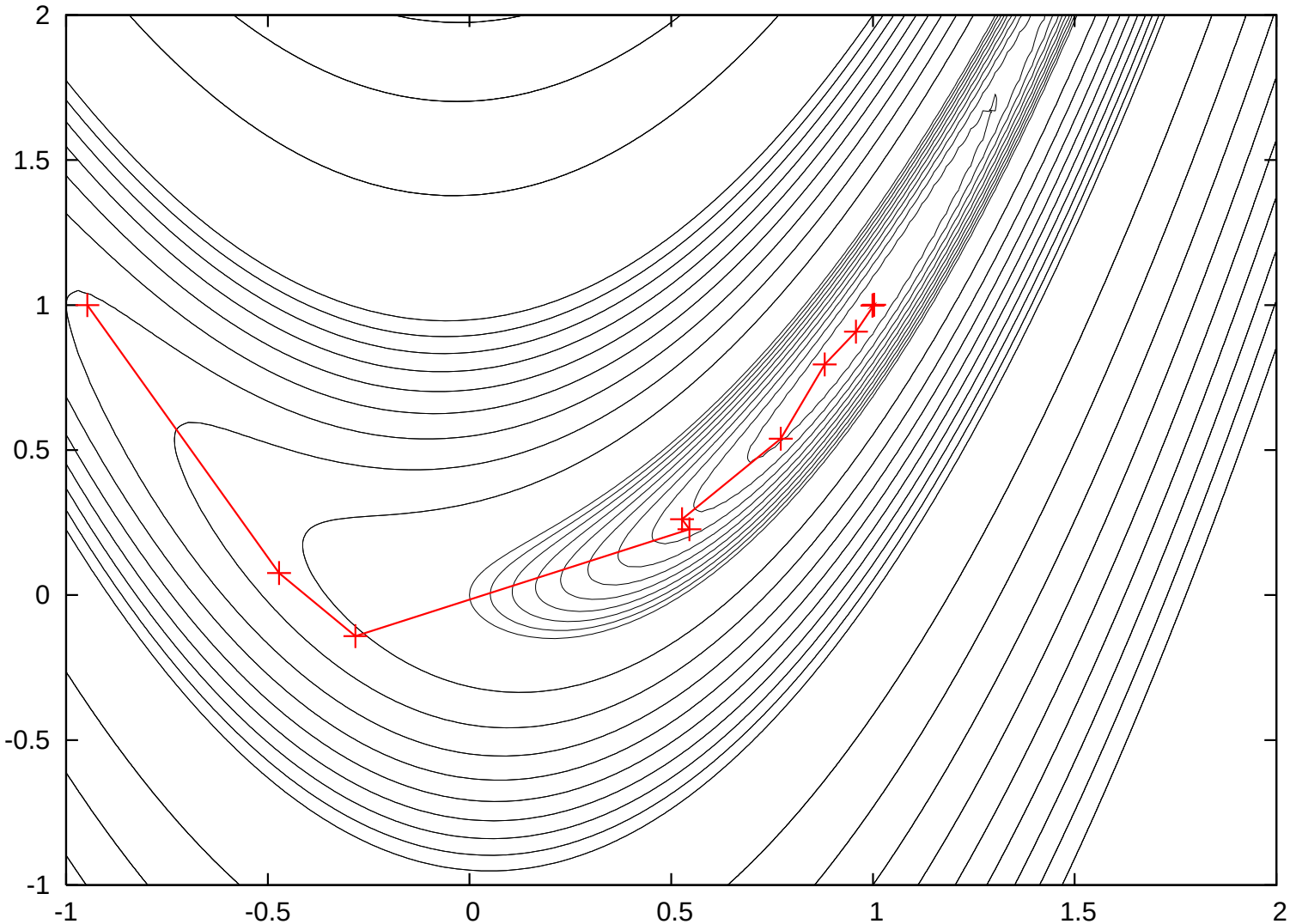
▷▷ GSL Library

Conjugate Gradient



▷▷ GSL Library

BFGS



▷▷ GSL Library

Algos based on cost gradient

Previously algo are only based on the cost gradient $\nabla j(\psi)$:

- Steepest,
- conjugate gradient,
- DFP, BFGS

There are others

that also use the "sensitivity" of the state wrt parameters (cf Lecture 2)

Cost of the kind

$$j(\psi) := \mathcal{J}(u) = \int_{\mathcal{S}} (u - u_d)^2 \, \mathrm{d}s$$

Definition (Directional derivative)

$u'(\psi;\delta\psi)$ is the derivative of the state $u(\psi)$ at the point ψ in the direction $\delta\psi$:

$$u'(\psi;\delta\psi) := \lim_{\epsilon\to 0} \frac{u(\psi+\epsilon\delta\psi)-u(\psi)}{\epsilon}$$

then the directional derivative of the cost function writes :

$$j'(\psi;\delta\psi) = \left(\mathcal{J}'(u), u'(\psi;\delta\psi)\right),$$

where

$$j'(\psi;\delta\psi) = (\nabla j(\psi), \delta\psi)$$

Gauss–Newton

Second derivative

the second derivative of $j(\psi)$ at the point ψ in the directions $\delta\psi$ and $\delta\phi$ is given by :

$$j''(\psi;\delta\psi,\delta\phi) = \left(\mathcal{J}'(u), u''(\psi;\delta\psi,\delta\phi)\right) + \left(\left(\mathcal{J}''(u), u'(\psi;\delta\psi)\right), u'(\psi;\delta\phi)\right).$$

Neglecting the second-order term (this is actually the Gauss–Newton approach), we have :

$$j''(\psi;\delta\psi,\delta\phi) \approx \left(\left(\mathcal{J}''(u), u'(\psi;\delta\psi)\right), u'(\psi;\delta\phi)\right).$$

Gauss–Newton

$$S^t S \delta\psi^k = -\nabla j(\psi^k)$$

Matrix $S^t S$ usually badly conditionned

Damp the system

Levenberg–Marquardt

$$\left[S^t S + \ell I\right] \delta\psi^k = -\nabla j(\psi^k)$$

or better :

$$\left[S^t S + \ell \mathrm{diag}(S^t S)\right] \delta\psi^k = -\nabla j(\psi^k)$$

Remark

Note that $\ell \to 0$ yields the Gauss–Newton algorithm while ℓ bigger gives an approximation of the steepest descent gradient algorithm. In practice, the parameter ℓ may be adjusted at each iteration.

Remark

when $\dim \psi$ is high $\to$ prefer gradient-based methods

Opt. method / Gradient method

Steepest, conjugate-grad., BFGS, DFP, ...	Newton	Gauss–Newton, Levenberg–Marquardt, ...
$u \leftarrow \mathcal{R}(u, \psi) = 0$ $j \leftarrow u$ $\nabla j \leftarrow \left\{ \begin{array}{l} \text{Forward diff.} \\ \text{or} \\ \text{Adjoint state} \end{array} \right.$	$u \leftarrow \mathcal{R}(u, \psi) = 0$ $j \leftarrow u$ $\nabla j \leftarrow \left\{ \begin{array}{l} \text{Forward diff.} \\ \text{or} \\ \text{Adjoint state} \end{array} \right.$ $\nabla^2 j$ (complicated)	$u \leftarrow \mathcal{R}(u, \psi) = 0$ $j \leftarrow u$ $S^t S \leftarrow S \leftarrow u'$ (Forw. diff.)

Contents

We compute :

- The state :

$$\mathcal{R}(u, \psi) = 0$$

- The cost :

$$j(\psi) := \mathcal{J}(u)$$

We search

$$\inf j(\psi)$$

We need

$$\nabla j(\psi)$$

Definition

defining $\left| \begin{array}{c} u'(\psi;\delta\psi) \\ j'(\psi;\delta\psi) \end{array} \right|$ the derivative of the $\left| \begin{array}{c} \text{state} \\ \text{cost} \end{array} \right|$ at the point ψ in the direction $\delta\psi$ as :

$$u'(\psi;\delta\psi) := \lim_{\epsilon \to 0} \frac{u(\psi + \epsilon\delta\psi) - u(\psi)}{\epsilon}$$

$$j'(\psi;\delta\psi) := \lim_{\epsilon \to 0} \frac{j(\psi + \epsilon\delta\psi) - j(\psi)}{\epsilon}$$

then the directional derivative of the cost function writes :

$$j'(\psi;\delta\psi) = \left(\mathcal{J}'(u), u'(\psi;\delta\psi)\right),$$

where

$$j'(\psi;\delta\psi) = (\nabla j(\psi), \delta\psi)$$

methods

- Foward differentiation : Finite Differences
- Foward differentiation of the equations
- Adjoint

Finite difference

Approximation of ∇j

whole canonical base of ψ, that is $\delta\psi = \delta\psi_1, \delta\psi_2, \ldots, \delta\psi_{\dim \psi}$.

For the i^{th} component, we have :

$$(\nabla j(\psi))_i = (\nabla j(\psi), \delta\psi_i) \approx \frac{j(\psi + \epsilon\delta\psi_i) - j(\psi)}{\epsilon}.$$

Often

in order to perform the same relative perturbation on all components ψ_i, one uses $\epsilon_i \leftarrow \varepsilon\psi_i$, where the scalar ε is fixed.

$$(\nabla j(\psi))_i \leftarrow \frac{j(\psi^p + \varepsilon\psi_i\delta\psi_i) - j(\psi^p)}{\varepsilon\psi_i}$$

Finite difference

Algorithm 4: The finite difference algorithm to compute the gradient of the cost function

Set the length ε;

At iteration p, compute the state $u(\psi^p)$, compute $j(\psi^p)$;

foreach $i = 1, \dots, \dim \psi$ **do**

  Compute the cost $j(\psi^p + \varepsilon\psi_i \delta\psi_i)$;

  Set the gradient $(\nabla j(\psi))_i \leftarrow \dfrac{j(\psi^p + \varepsilon\psi_i \delta\psi_i) - j(\psi^p)}{\varepsilon\psi_i}$

Integrate the gradient within the optimization methods that do not rely on the sensitivities (conjugate gradient or BFGS for instance among the presented methods)

Remark (The tuning parameter ε)

has to be chosen within a region where variables depend roughly linearly on ε.

- for too small values, the round-off errors dominate
- for too high values one gets the nonlinear behavior

Remark (Expensive)

At each iteration p, one needs $\dim \psi$ integrations of $R(u, \psi) = 0$ to get ∇j

Finite difference

FD to approach $u'(\psi;\delta\psi_i)$ usable in G–N and L–M algos

Algorithm 5: The finite difference algorithm to compute the gradient of the cost function and the sensitivities

Set the step ε;
At iteration p, compute the state $u(\psi^p)$, compute $j(\psi^p)$;
foreach $i = 1, \ldots, \dim \psi$ **do**

Compute the perturbed state $u(\psi^p + \varepsilon\psi_i\delta\psi_i)$ and the cost $j(\psi^p + \varepsilon\psi_i\delta\psi_i)$;

Set the state sensitivity $u'(\psi;\delta\psi_i) \leftarrow \dfrac{u(\psi^p + \varepsilon\psi_i\delta\psi_i) - u(\psi^p)}{\varepsilon\psi_i}$;

Set the gradient $(\nabla j, \delta\psi_i)$ with

- either $(\mathcal{J}'(u), u'(\psi;\delta\psi_i))$
- or as in previous algorithm with $\dfrac{j(\psi^p + \varepsilon\psi_i\delta\psi_i) - j(\psi^p)}{\varepsilon\psi_i}$.

Integrate the gradient within the optimization methods that

- do not rely on the sensitivities (steepest, conjugate gradient, BFGS, etc.)
- or within optimization methods that do rely on the sensitivities (Gauss–Newton, Levenberg–Marquardt, etc.).

Forward differentiation

Computation of $u'(\psi;\delta\psi)$ through differentiation of $\mathcal{R}(u,\psi)$:

$$\mathcal{R}(u,\psi) = 0$$

$$\mathcal{R}'_u(u,\psi)u' + \mathcal{R}'_\psi(u,\psi)\delta\psi = 0$$

Then

$$(\nabla j(\psi), \delta\psi) = j'(\psi;\delta\psi) = \left(\mathcal{J}'(u), u'(\psi;\delta\psi)\right)$$

Forward differentiation

Algorithm 6: The forward differentiation algorithm to compute the cost gradient and the state sensitivities

At iteration p, solve (iteratively) $\mathcal{R}(u, \psi^p) = 0$;

Compute $j(\psi^p)$ and save the linear tangent matrix $\mathcal{R}'_u(u, \psi^p)$;

foreach $i = 1, \ldots, \dim \psi$ **do**

Solve

$$\mathcal{R}'_u(u, \psi)u' + \mathcal{R}'_\psi(u, \psi)\delta\psi_i = 0$$

Set

$$(\nabla j, \delta\psi_i) = \left(\mathcal{J}'(u), u'(\psi; \delta\psi_i)\right)$$

Integrate the gradient within the optimization methods that

- do not rely on the sensitivities (steepest, conjugate gradient, BFGS, etc.)
- or within optimization methods that do rely on the sensitivities (Gauss–Newton, Levenberg–Marquardt, etc.).

Forward differentiation

Example

problem statement

Looking for a $\psi \leftarrow h$ in transient heat conduction
measurements u_d,
$\inf j(\lambda)$

equations

$$\begin{cases} C\dot{T} - \nabla \cdot (\lambda \nabla T) = f & \boldsymbol{x} \in \Omega,\ t \in \mathcal{I} \\ T = T_0 & \boldsymbol{x} \in \Omega,\ t = 0 \\ \nabla T \cdot \boldsymbol{n} = 0 & \boldsymbol{x} \in \partial\Omega_1,\ t \in \mathcal{I} \\ \lambda \nabla T \cdot \boldsymbol{n} = -h\,(T - T_\infty) & \boldsymbol{x} \in \partial\Omega_2,\ t \in \mathcal{I} \\ \lambda \nabla T \cdot \boldsymbol{n} = -\varepsilon\sigma\left(T^4 - T_\infty^4\right) & \boldsymbol{x} \in \partial\Omega_3,\ t \in \mathcal{I} \end{cases}$$

Forward differentiation

Example

PDE

$$C\dot{T} - \nabla \cdot (\lambda \nabla T) - f = 0$$

if linear

$$C\dot{T}' - \lambda \Delta T' = 0$$

if non linear : $C(T)$, $\lambda(T)$

$$\frac{\partial}{\partial t}(CT') - \Delta(\lambda T') = 0$$

Forward differentiation

Example

IC

$$T' = 0$$

null flux BC

$$\nabla T' \cdot \boldsymbol{n} = 0$$

term $\lambda \nabla T \cdot \boldsymbol{n}$

$$\lambda'(T) T' \nabla T \cdot \boldsymbol{n} + \lambda \nabla T' \cdot \boldsymbol{n} = \left(T' \nabla \lambda + \lambda \nabla T' \right) \cdot \boldsymbol{n} = \nabla (\lambda T') \cdot \boldsymbol{n}$$

term $h(T - T_\infty)$

$$hT'$$

term $\varepsilon\sigma \left(T^4 - T_\infty^4 \right)$

$$4\varepsilon\sigma T^3 T'$$

Forward differentiation

Example

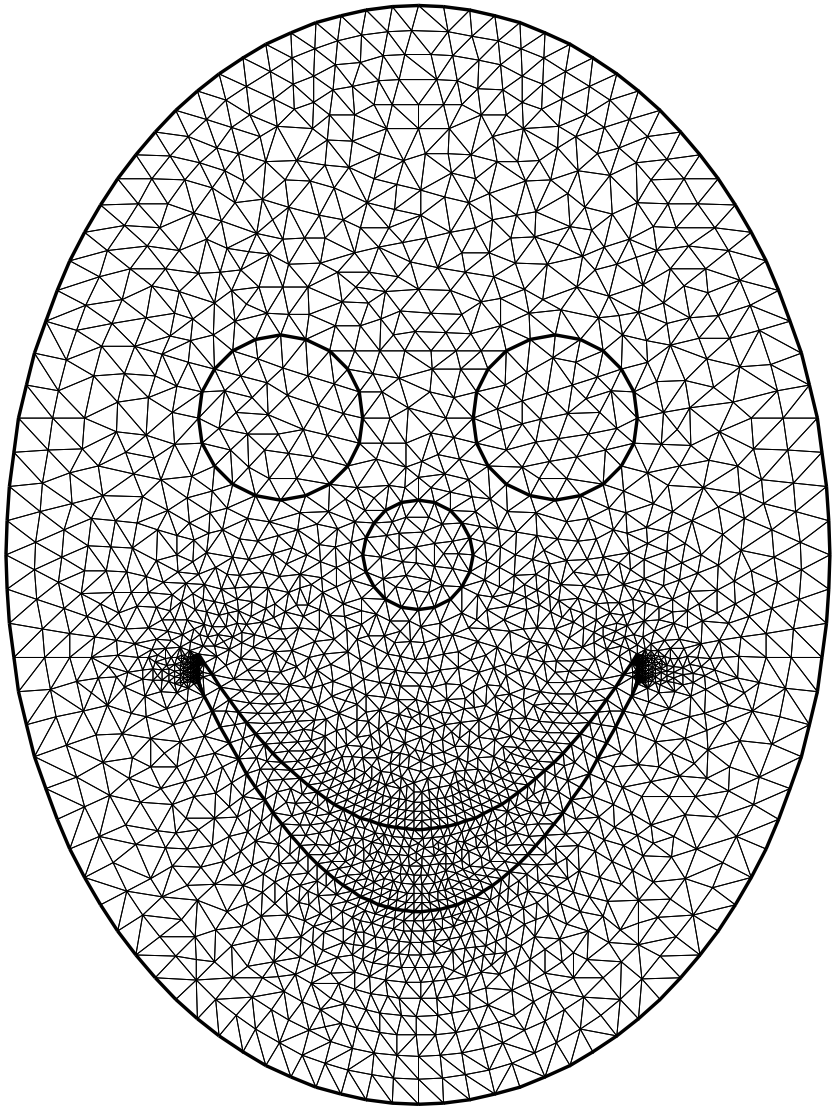
State equations (recall)

$$\begin{cases} C\dot{T} - \nabla \cdot (\lambda \nabla T) = f & \boldsymbol{x} \in \Omega,\ t \in \mathcal{I} \\ T' = 0 & \boldsymbol{x} \in \Omega,\ t = 0 \\ \nabla T \cdot \boldsymbol{n} = 0 & \boldsymbol{x} \in \partial\Omega_1,\ t \in \mathcal{I} \\ \lambda \nabla T \cdot \boldsymbol{n} = -h\,(T - T_\infty) & \boldsymbol{x} \in \partial\Omega_2,\ t \in \mathcal{I} \\ \lambda \nabla T \cdot \boldsymbol{n} = -\varepsilon\sigma\left(T^4 - T_\infty^4\right) & \boldsymbol{x} \in \partial\Omega_3,\ t \in \mathcal{I} \end{cases}$$

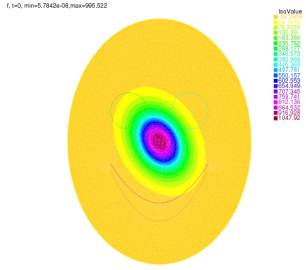
If $\psi \leftarrow h$ – Differentiated equations

$$\begin{cases} \frac{\partial}{\partial t}(CT') - \nabla \cdot (\nabla(\lambda T')) = 0 & \boldsymbol{x} \in \Omega,\ t \in \mathcal{I} \\ T = T_0 & \boldsymbol{x} \in \Omega,\ t = 0 \\ \nabla T' \cdot \boldsymbol{n} = 0 & \boldsymbol{x} \in \partial\Omega_1,\ t \in \mathcal{I} \\ \nabla(\lambda T') \cdot \boldsymbol{n} = -hT' - \delta h(T - T_\infty) & \boldsymbol{x} \in \partial\Omega_2,\ t \in \mathcal{I} \\ \nabla(\lambda T') \cdot \boldsymbol{n} = -4\varepsilon\sigma T^3 T' & \boldsymbol{x} \in \partial\Omega_3,\ t \in \mathcal{I} \end{cases}$$

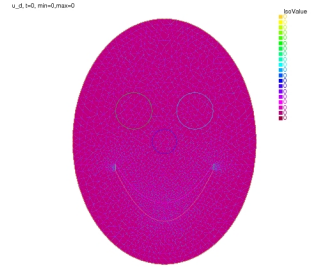
The philisophy with 4 parameters



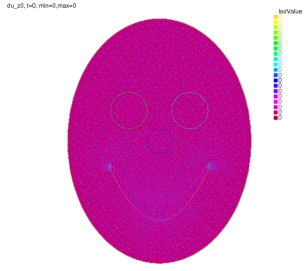
f



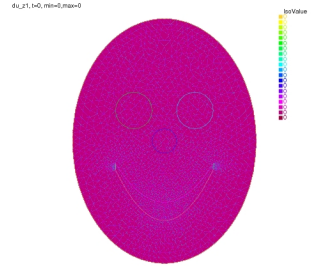
$u(\lambda)$



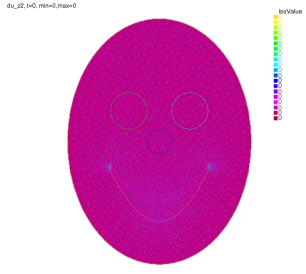
$u'(\lambda;\delta\lambda_1)$



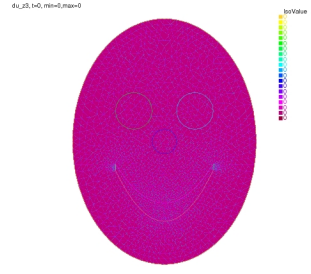
$u'(\lambda;\delta\lambda_2)$



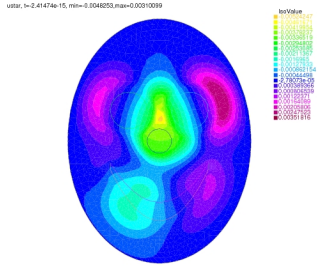
$u'(\lambda;\delta\lambda_3)$



$u'(\lambda;\delta\lambda_4)$



u^*



Forward differentiation of PDE

Advantages w.r.t. FD

- Exact computation
- Less CPU comsuming (e.g. parameters depend on state)

But still expensive

- Still have $\dim \psi$ resolutions

Adjoint state method

Main featurzs

- Only one “adjoint” system to access the full gradient
- Theory of optimal control, 70’s
- Lots of ways for the method presentation

Adjoint state method

Computation of $u'(\psi;\delta\psi)$ through differentiation of $\mathcal{R}(u,\psi)$:

$$\mathcal{R}'_u(u,\psi)u' + \mathcal{R}'_\psi(u,\psi)\delta\psi = 0 \tag{a}$$

$$(\nabla j(\psi),\delta\psi) = j'(\psi;\delta\psi) = \left(\mathcal{J}'(u), u'(\psi;\delta\psi)\right) \tag{b}$$

Instead, we search :

$$(\nabla j(\psi),\delta\psi) = \left(\mathcal{R}'_\psi(u,\psi)\delta\psi, u^*\right) \tag{c}$$

With (a) and (c) :

$$(\nabla j(\psi),\delta\psi) = -\left(\mathcal{R}'_u(u,\psi)u', u^*\right) \tag{d}$$

Identifying (b) and (d) :

$$\left(\mathcal{J}'(u), u'(\psi;\delta\psi)\right) = -\left(\mathcal{R}'_u(u,\psi)u', u^*\right) = -\left(\mathcal{R}^*(u,\psi)u^*, u'\right) \tag{e}$$

Adjoint state method

Recalls for previous slide

We search :

$$(\nabla j(\psi), \delta\psi) = \left(\mathcal{R}'_\psi(u,\psi)\delta\psi, u^*\right) \tag{c}$$

We have also :

$$(\nabla j(\psi), \delta\psi) = -\left(\mathcal{R}'_u(u,\psi)u', u^*\right) \tag{d}$$

We identify :

$$\left(\mathcal{J}'(u), u'(\psi;\delta\psi)\right) = -\left(\mathcal{R}'_u(u,\psi)u', u^*\right) = -\left(\mathcal{R}^*(u,\psi)u^*, u'\right) \tag{e}$$

Adjoint equation (from (e)) :

$$\mathcal{R}^*(u,\psi)u^* + \mathcal{J}'(u)$$

Gradient (c) :

$$(\nabla j(\psi), \delta\psi) = \left(\mathcal{R}'_\psi(u,\psi)\delta\psi, u^*\right)$$

En résumé

1. Le problème direct :
$$\mathcal{R}(u, \psi) = 0;$$
2. Le problème adjoint :
$$\mathcal{R}^*(u, \psi)u^* + \mathcal{J}'(u)$$
3. Le gradient du coût :
$$(\nabla j(\psi), \delta\psi) = \left(\mathcal{R}'_\psi(u, \psi)\delta\psi, u^*\right)$$

Example

Case of (linear) set of ODE

$$\begin{aligned} \mathscr{C}\dot{u} - \mathscr{B} &= 0 \qquad \text{for } t \in \mathcal{I} \\ u &= u_0 \qquad \text{for } t = 0, \end{aligned} \tag{$*$}$$

Injecting $(*)$ into $(R'_u(u,\psi)u', u^*) + (\mathcal{J}'(u), u'(\psi;\delta\psi)) = 0.$:

$$\left(\mathscr{C}\frac{d}{dt}u', u^*\right) + \left(\mathcal{J}'(u), u'(\psi;\delta\psi)\right) = 0$$

transpose operator $\mathscr{C}$

$$\left(\frac{d}{dt}u', \mathscr{C}^* u^*\right) + \left(\mathcal{J}', u'\right) = 0$$

$$\left(\frac{d}{dt}u', \mathscr{C}^* u^*\right) + \left(\mathscr{J}', u'\right) = 0$$

int. by parts

$$-\left(u', \mathscr{C}^* \frac{d}{dt}u^*\right) + \left[\langle u', \mathscr{C}^* u^* \rangle\right]_0^{t_f} + \left(\mathscr{J}'(u), u'(\psi; \delta\psi)\right) = 0$$

Eventually

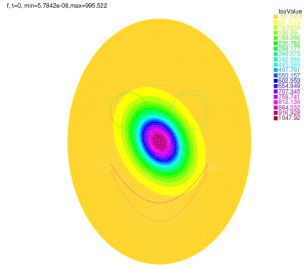
$$\begin{aligned} -\mathscr{C}^* \dot{u}^* + \mathscr{J}'(u) &= 0 \qquad && \text{for } t \in \mathcal{I} \\ u^* &= 0 && \text{for } t = t_f \ . \end{aligned}$$

Algorithm 7: The global optimization algorithm

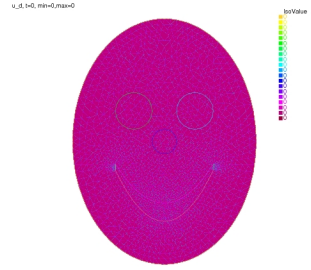
1. Integrate the cost function value through integration of the forward (maybe nonlinear) direct problem;
   Store all state variables to reconstruct the tangent matrix (or store the tangent matrix);
2. Integrate the backward linear adjoint problem, all matrices being possibly stored or recomputed from stored state variables
3. Compute the cost function gradient;
   Compute the direction of descent
4. Solve the line research algorithm through several integrations of the nonlinear direct model.

1. Forward finite difference method :
   - $(\dim \psi + 1)$ nonlinear resolution of
   $$\mathcal{R}(u, \psi) = 0$$
2. Forward differentiation method :
   - 1 nonlinear resolution of
   $$\mathcal{R}(u, \psi) = 0$$
   - $\dim \psi$ resolution of the linear tangent model
   $$\mathcal{R}'_u(u, \psi)u' + \mathcal{R}'_\psi(u, \psi)\delta\psi = 0$$
3. Adjoint state method :
   - 1 nonlinear resolution of
   $$\mathcal{R}(u, \psi) = 0$$
   - 1 resolution of the adjoint of the tangent model :
   $$\mathcal{R}^*(u, \psi)u^* + \mathcal{J}'(u) = 0$$

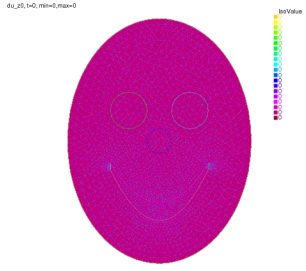
f



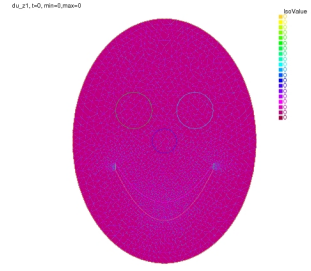
$u(\lambda)$



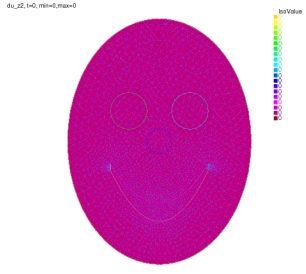
$u'(\lambda;\delta\lambda_1)$



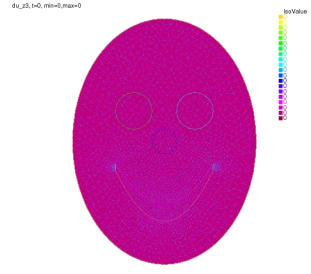
$u'(\lambda;\delta\lambda_2)$



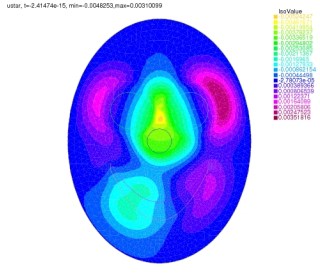
$u'(\lambda;\delta\lambda_3)$

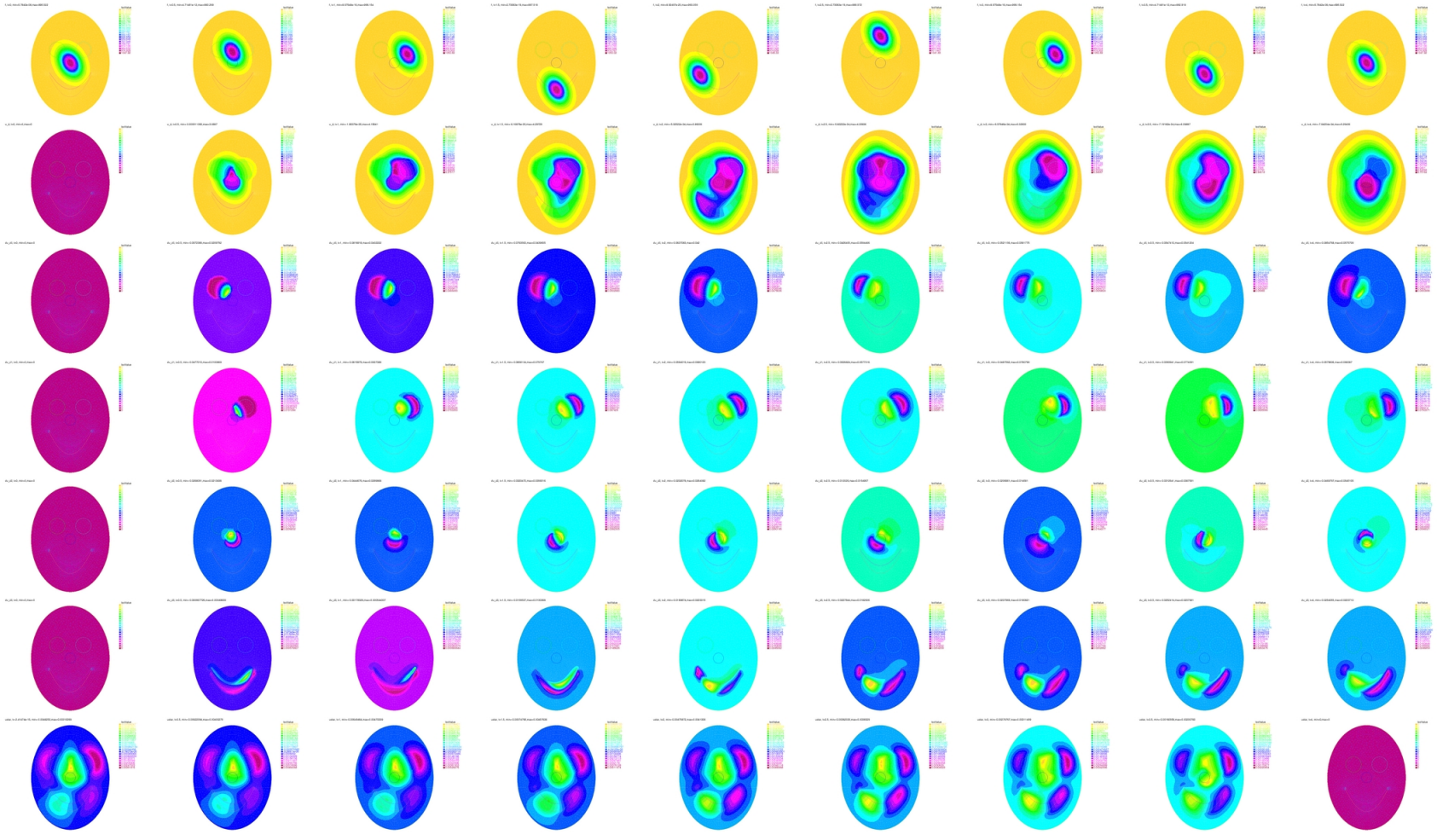


$u'(\lambda;\delta\lambda_4)$



u^*





Another example – another method

State equations

$$\begin{cases} C\dot{T} - \nabla \cdot (\lambda \nabla T) = f & \boldsymbol{x} \in \Omega,\ t \in \mathcal{I} \\ T = 0 & \boldsymbol{x} \in \Omega,\ t = 0 \\ \nabla T \cdot \boldsymbol{n} = 0 & \boldsymbol{x} \in \partial\Omega_1,\ t \in \mathcal{I} \\ \lambda \nabla T \cdot \boldsymbol{n} = -h\,(T - T_\infty) & \boldsymbol{x} \in \partial\Omega_2,\ t \in \mathcal{I} \\ \lambda \nabla T \cdot \boldsymbol{n} = -\varepsilon\sigma\left(T^4 - T_\infty^4\right) & \boldsymbol{x} \in \partial\Omega_3,\ t \in \mathcal{I} \end{cases}$$

If $\psi \leftarrow h$ – Differentiated equations

$$\begin{cases} \frac{\partial}{\partial t}(CT') - \nabla \cdot (\nabla(\lambda T')) = 0 & \boldsymbol{x} \in \Omega,\ t \in \mathcal{I} \\ T = T_0 & \boldsymbol{x} \in \Omega,\ t = 0 \\ \nabla T' \cdot \boldsymbol{n} = 0 & \boldsymbol{x} \in \partial\Omega_1,\ t \in \mathcal{I} \\ \nabla(\lambda T') \cdot \boldsymbol{n} = -hT' - \delta h(T - T_\infty) & \boldsymbol{x} \in \partial\Omega_2,\ t \in \mathcal{I} \\ \nabla(\lambda T') \cdot \boldsymbol{n} = -4\varepsilon\sigma T^3 T' & \boldsymbol{x} \in \partial\Omega_3,\ t \in \mathcal{I} \end{cases}$$

The Lagrange function (we take $\psi \leftarrow h$)

$$\begin{aligned}\mathscr{L}(T,\{T^*,\eta,\gamma,\xi,\varpi\},h) = \quad & \mathcal{J}(T) + \left(C\tfrac{\partial T}{\partial t} - \nabla\cdot(\lambda\nabla T) - f, T^*\right)_{L_2(0,T;L_2(\Omega))} \\ & + \left(T - T_0, \eta\right)_{L_2(\Omega)}(t=0) \\ & + \left(\nabla T\cdot\boldsymbol{n}, \xi\right)_{L_2(0,T;L_2(\partial\Omega_1))} \\ & + \left(\lambda\nabla T\cdot\boldsymbol{n} + h\left(T - T_\infty\right), \gamma\right)_{L_2(0,T;L_2(\partial\Omega_2))} \\ & + \left(\lambda\nabla T\cdot\boldsymbol{n} + \varepsilon\sigma\left(T^4 - T_\infty^4\right), \varpi\right)_{L_2(0,T;L_2(\partial\Omega_3))}\end{aligned}$$

The differentiated Lagrange function with respect to h is the direction δh is :

$$\begin{aligned}\mathscr{L}_h'(\,\cdot\,)\delta h = \quad & \left(\mathcal{J}'(T), T'\right)_{L_2(0,T;L_2(\Omega))} \\ & + \left(\tfrac{\partial(CT')}{\partial t} - \nabla\cdot\left(\nabla(\lambda T')\right), T^*\right)_{L_2(0,T;L_2(\Omega))} \\ & + \left(T', \eta\right)_{L_2(\Omega)}(t=0) \\ & + \left(\nabla T'\cdot\boldsymbol{n}, \xi\right)_{L_2(0,T;L_2(\partial\Omega_1))} \\ & + \left(\nabla(\lambda T')\cdot\boldsymbol{n} + hT' + \delta h(T - T_\infty), \gamma\right)_{L_2(0,T;L_2(\partial\Omega_2))} \\ & + \left(\nabla(\lambda T')\cdot\boldsymbol{n} + 4\varepsilon\sigma T^3 T', \varpi\right)_{L_2(0,T;L_2(\partial\Omega_3))}\end{aligned}$$

The term related to the PDE

$$\left(\frac{\partial(CT')}{\partial t} - \nabla\cdot\left(\nabla(\lambda T')\right), T^*\right)_{L_2(0,T;L_2(\Omega))}$$

We use

$$\left(\frac{\partial(CT')}{\partial t}, T^*\right)_{L_2(0,T;L_2(\Omega))} = \left(T', -C\frac{\partial T^*}{\partial t}\right)_{L_2(0,T;L_2(\Omega))} + \left(CT', T^*\right)_{L_2(\Omega)}(t = t_f)$$

We use

$$\begin{aligned}(\Delta(\lambda T'), T^*)_{L_2(0,T;L_2(\Omega))} = \quad & (\lambda\Delta T^*, T')_{L_2(0,T;L_2(\Omega))} \\ & + (\lambda T^*, \nabla T'\cdot\boldsymbol{n})_{L_2(0,T;L_2(\partial\Omega))} \\ & + (\nabla\lambda\cdot\boldsymbol{n}T^*, T')_{L_2(0,T;L_2(\partial\Omega))} \\ & - (\lambda\nabla T^*\cdot\boldsymbol{n}, T')_{L_2(0,T;L_2(\partial\Omega))}\end{aligned}$$

. . . and do the same for all BC

We bring together similar terms to get :

$$\left(\mathscr{L}_h'(T,\{T^*,\eta,\gamma,\xi,\varpi\},h),\delta h\right) = \text{(I)} + \text{(II)} + \text{(III)} + \text{(IV)} + \text{(V)}$$

where

$$\begin{aligned}
\text{(I)} :=& \left(\mathcal{J}'(T),T'\right)_{L_2(0,T;L_2(\Omega))} + \left(\delta h(T-T_\infty),\gamma\right)_{L_2(0,T;L_2(\partial\Omega_2))} \\
\text{(II)} :=& \left(-C\frac{\partial T^*}{\partial t} - \lambda\Delta T^*, T'\right)_{L_2(0,T;L_2(\Omega))} \\
\text{(III)} :=& \left(\nabla\lambda\cdot\boldsymbol{n}T^* - \lambda\nabla T^*\cdot\boldsymbol{n}, T'\right)_{L_2(0,T;L_2(\partial\Omega))} + \left(\nabla\lambda\cdot\boldsymbol{n}\gamma + h\gamma, T'\right)_{L_2(0,T;L_2(\partial\Omega_2))} \\
&+ \left(\nabla\lambda\cdot\boldsymbol{n}\varpi, T'\right)_{L_2(0,T;L_2(\partial\Omega_3))} + \left(4\epsilon\sigma T^3\varpi, T'\right)_{L_2(0,T;L_2(\partial\Omega_3))} \\
\text{(IV)} :=& \left(\lambda T^*, \nabla T'\cdot\boldsymbol{n}\right)_{L_2(0,T;L_2(\partial\Omega))} + \left(\lambda\gamma, \nabla T'\cdot\boldsymbol{n}\right)_{L_2(0,T;L_2(\partial\Omega_2))} \\
&+ \left(\xi, \nabla T'\cdot\boldsymbol{n}\right)_{L_2(0,T;L_2(\partial\Omega_1))} + \left(\lambda\varpi, \nabla T'\cdot\boldsymbol{n}\right)_{L_2(0,T;L_2(\partial\Omega_3))} \\
\text{(V)} :=& \left(CT^*, T'\right)_{L_2(\Omega)}(t=t_f)
\end{aligned}$$

exmaple of the radiative BC (on $\partial\Omega_3$)

$$
\begin{aligned}
\text{(I)} &:= \left(\mathcal{J}'(T), T'\right)_{L_2(0,T;L_2(\Omega))} + \left(\delta h(T - T_\infty), \gamma\right)_{L_2(0,T;L_2(\partial\Omega_2))} \\
\text{(II)} &:= \left(-C\frac{\partial T^*}{\partial t} - \lambda\Delta T^*, T'\right)_{L_2(0,T;L_2(\Omega))} \\
\text{(III)} &:= \left(\nabla\lambda\cdot\boldsymbol{n}T^* - \lambda\nabla T^*\cdot\boldsymbol{n}, T'\right)_{L_2(0,T;L_2(\partial\Omega))} + \left(\nabla\lambda\cdot\boldsymbol{n}\gamma + h\gamma, T'\right)_{L_2(0,T;L_2(\partial\Omega_2))} \\
&\quad + \left(\nabla\lambda\cdot\boldsymbol{n}\varpi, T'\right)_{L_2(0,T;L_2(\partial\Omega_3))} + \left(4\epsilon\sigma T^3\varpi, T'\right)_{L_2(0,T;L_2(\partial\Omega_3))} \\
\text{(IV)} &:= \left(\lambda T^*, \nabla T'\cdot\boldsymbol{n}\right)_{L_2(0,T;L_2(\partial\Omega))} + \left(\lambda\gamma, \nabla T'\cdot\boldsymbol{n}\right)_{L_2(0,T;L_2(\partial\Omega_2))} \\
&\quad + \left(\xi, \nabla T'\cdot\boldsymbol{n}\right)_{L_2(0,T;L_2(\partial\Omega_1))} + \left(\lambda\varpi, \nabla T'\cdot\boldsymbol{n}\right)_{L_2(0,T;L_2(\partial\Omega_3))} \\
\text{(V)} &:= \left(CT^*, T'\right)_{L_2(\Omega)}(t = t_f)
\end{aligned}
$$

We choose (for vanishing) :

$$
\left(\lambda\varpi + \lambda T^*, \nabla T'\cdot\boldsymbol{n}\right)_{L_2(0,T;L_2(\partial\Omega_3))} = 0 \Longrightarrow \varpi + T^*|_{\partial\Omega_4} = 0
$$

$$
\left(\nabla\lambda\cdot\boldsymbol{n}\varpi + 4\epsilon\sigma T^3\varpi + \nabla\lambda\cdot\boldsymbol{n}T^* - \lambda\nabla T^*\cdot\boldsymbol{n}, T'\right)_{L_2(0,T;L_2(\partial\Omega_4))} = 0
$$

$$
\Longrightarrow -\lambda\nabla T^*\cdot\boldsymbol{n} - 4\epsilon\sigma T^3T^*\big|_{\partial\Omega_3} = 0
$$

We do again for all BC

and combine the relationships

to find that if

$$\begin{cases} -C\frac{\partial T^*}{\partial t} - \lambda\Delta T^* + \mathcal{J}'(T) = 0 & \boldsymbol{x} \in \Omega,\ t \in \mathcal{I} \\ T^* = 0 & \boldsymbol{x} \in \Omega,\ t = t_f \\ \nabla T^* \cdot \boldsymbol{n} = 0 & \boldsymbol{x} \in \partial\Omega_1,\ t \in \mathcal{I} \\ -\lambda\nabla T^* \cdot \boldsymbol{n} = hT^* & \boldsymbol{x} \in \partial\Omega_2,\ t \in \mathcal{I} \\ -\lambda\nabla T^* \cdot \boldsymbol{n} = 4\epsilon\sigma T^3 T^* & \boldsymbol{x} \in \partial\Omega_3,\ t \in \mathcal{I} \end{cases}$$

Then the cost gradient is

$$\nabla j = -\left(T - T_\infty, T^*\right)_{L_2(0,T;L_2(\partial\Omega_1))}.$$

NB : State :

$$\begin{cases} C\dot{T} - \nabla\cdot(\lambda\nabla T) = f & \boldsymbol{x} \in \Omega,\ t \in \mathcal{I} \\ T' = 0 & \boldsymbol{x} \in \Omega,\ t = 0 \\ \nabla T \cdot \boldsymbol{n} = 0 & \boldsymbol{x} \in \partial\Omega_1,\ t \in \mathcal{I} \\ \lambda\nabla T \cdot \boldsymbol{n} = -h\left(T - T_\infty\right) & \boldsymbol{x} \in \partial\Omega_2,\ t \in \mathcal{I} \\ \lambda\nabla T \cdot \boldsymbol{n} = -\varepsilon\sigma\left(T^4 - T_\infty^4\right) & \boldsymbol{x} \in \partial\Omega_3,\ t \in \mathcal{I} \end{cases}$$

Adjoint features

- Sometimes difficult to write down
- CPU inexpensive
- Solve backward
- Coupled problems
- Choice of the inner product norm
-

and now :

Philip