Programming, numerics and optimization

Lecture C-3: Unconstrained optimization II

Łukasz Jankowski
ljank@ippt.pan.pl

Institute of Fundamental Technological Research
Room 4.32, Phone +22.8261281 ext. 428

May 11, 2021[1]

[1] Current version is available at http://info.ippt.pan.pl/~ljank.

Outline

Outline

Line search methods

The basic outline of all *line search* algorithms is:

- Select any feasible point $\mathbf{x}_0$.
- Having calculated points $\mathbf{x}_0$, $\mathbf{x}_1$, ..., $\mathbf{x}_k$, iteratively calculate the successive point $\mathbf{x}_{k+1}$:
  1. Choose the direction $\mathbf{d}_k$ of optimization.
  2. Starting from $\mathbf{x}_k$, perform a (usually approximate) 1D optimization in the direction of $\mathbf{d}_k$, that is find $s_k \in \mathbb{R}$ that sufficiently[2] decreases f_k and $|f_k'|$, where

$$f_k(s) = f(\mathbf{x}_k + s\,\mathbf{d}_k).$$

  Then, take the step

$$\mathbf{x}_{k+1} := \mathbf{x}_k + s_k\,\mathbf{d}_k.$$

  3. Check the stop conditions.

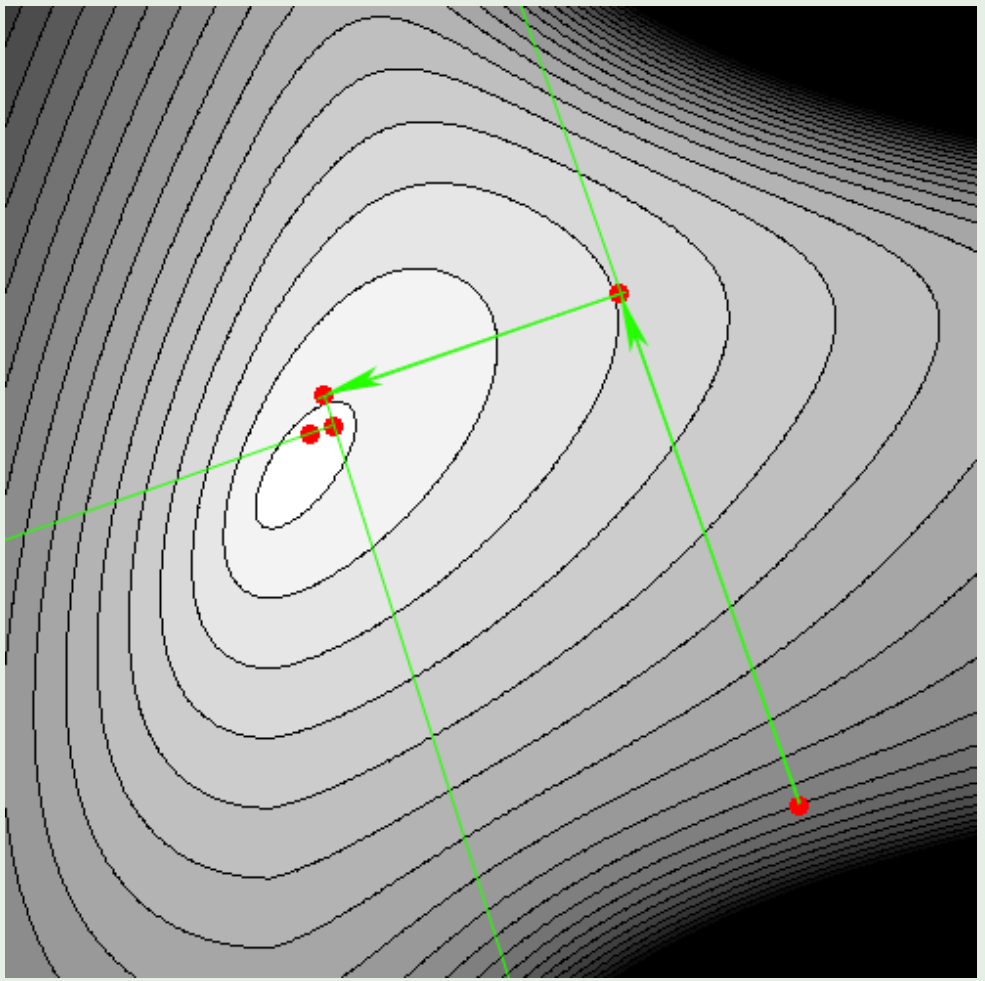
[2] Sufficiently, that is significantly enough to guarantee convergence to the minimum. Exact minimization is usually not necessary; it can be costly and sometimes it can even make the convergence slower.

Line search methods

1. Choose the direction of optimization.
2. Perform a (usually approximate) 1D optimization in that direction.
3. Check the stop conditions

Two problems:

1. Direction choice
2. Step size



Outline

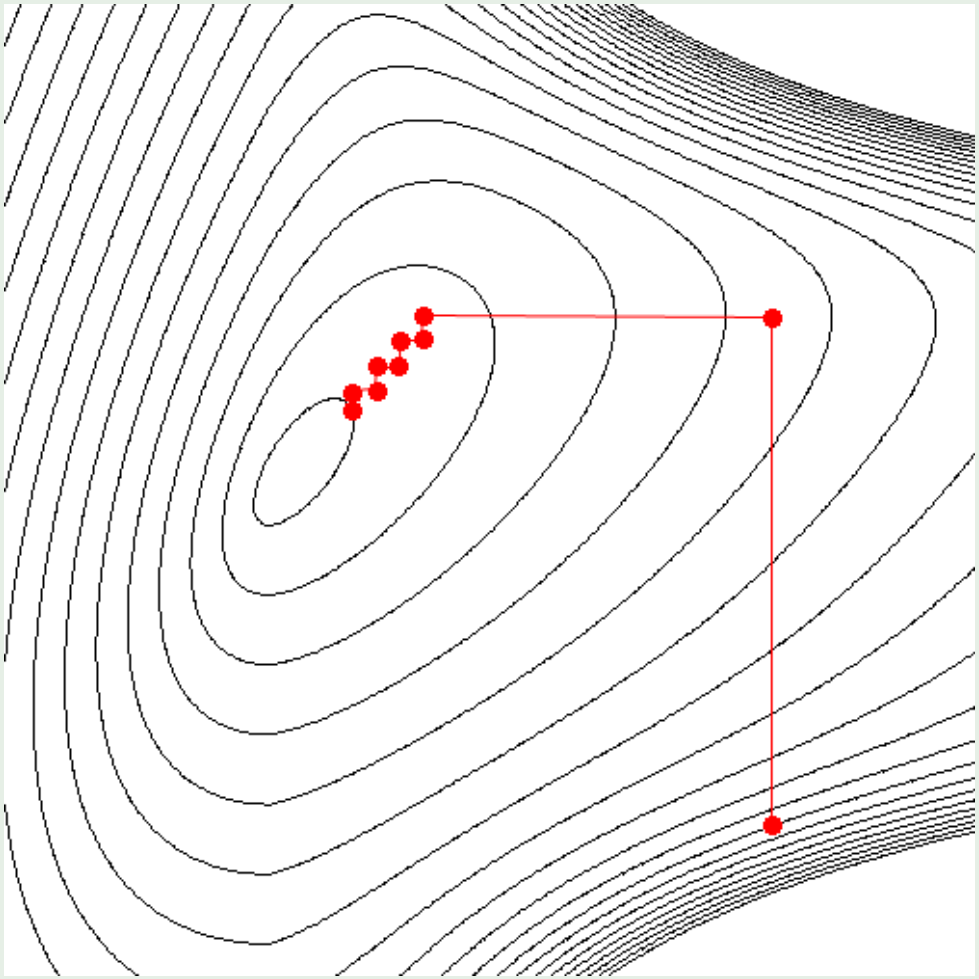
Zero order methods — Coordinate descent

The simplest method. All coordinate directions $\mathbf{e}_1$, ..., $\mathbf{e}_n$ are used sequentially,

$$\mathbf{d}_k := \mathbf{e}_{k \bmod n}.$$

- The simplest method
- Very slow or even nonconvergent.
- In 2D and with exact line minimizations equivalent to the steepest descent (besides the first step).

The set of the search directions can be periodically modified to include directions deemed to be more effective.



Zero order methods — Powell's direction set

Given $\mathbf{x}_0$,

1. Optimize along all the directions $\mathbf{e}_1, \ldots, \mathbf{e}_n$ and yield the points $\mathbf{x}_1, \ldots, \mathbf{x}_n$.
2. Modify the directions

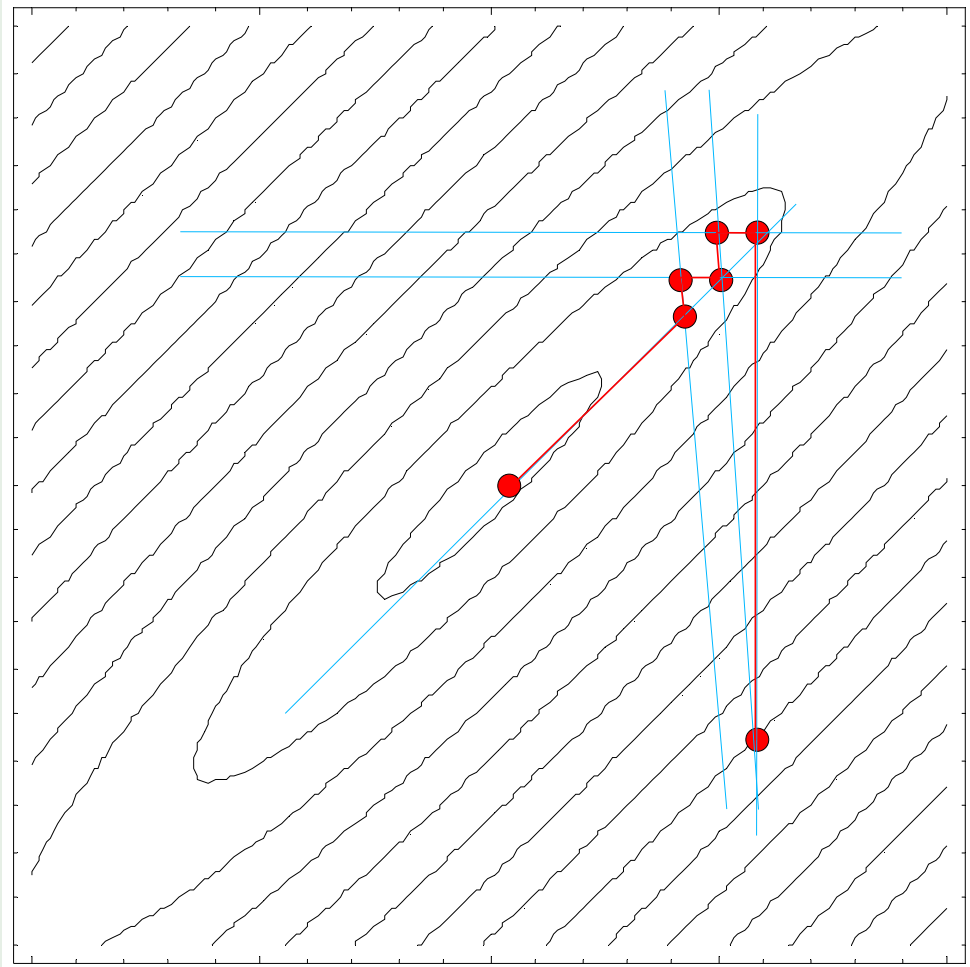
$$\mathbf{e}_1 := \mathbf{e}_2$$

$$\ldots$$

$$\mathbf{e}_{n-1} := \mathbf{e}_n$$

$$\mathbf{e}_n := \mathbf{x}_n - \mathbf{x}_0$$

3. Optimize along $\mathbf{e}_n$ and yield the point $\mathbf{x}_0$.
4. Repeat until stop conditions are satisfied.



Zero order methods — Powell's direction set

For a quadratic objective function and exact line minimizations, Powell's method generates a set of conjugate directions (besides the first step).

Iteratively generated directions tend to "fold up on each other" and become linearly dependent. Possible solutions:

- Every m iterations reset the directions to the original set.
- Every m iterations reset the directions to any orthogonal basis (making use of some of the already generated directions).
- When modifying the directions (step 2), instead of discarding the first direction $\mathbf{e}_1$, discard the direction of the largest decrease (since it is anyway a major component of the newly generated direction $\mathbf{e}_n$).

Zero order methods — Rosenbrock method

- The Rosenbrock method[3] involves approximate optimization that cycles over all the directions.
- The direction set is modified after each approximate optimization.

Single stage of the Rosenbrock method

Given $\mathbf{x}_0$.

1. Approximately optimize f using all the directions $\mathbf{e}_1, \ldots, \mathbf{e}_n$ and yield the points $\mathbf{x}_1, \ldots, \mathbf{x}_n$.
2. Modify the directions so that $\mathbf{e}_n := \mathbf{x}_n - \mathbf{x}_0$. Orthogonalize the resulting set.
3. Let $\mathbf{x}_0 := \mathbf{x}_n$.

[3] H.H. Rosenbrock. An Automatic Method for Finding the Greatest or Least Value of a Function. *The Computer Journal* **3**(3):175–184, 1960. Full text: `http://dx.doi.org/10.1093/comjnl/3.3.175`

Zero order methods — Rosenbrock method

Approximate optimization

1. Let Δs_i be an arbitrary initial step length in the ith direction.
2. Let $\alpha > 1$ and $\beta \in (0, 1)$ be two given constants (step elongation and step shortening).
3. Repeat

   for every direction $i = 1, \ldots, n$

   1. Make a step Δs_i in the ith direction.
   2. If successful (f not increased), then $\Delta s_i = \alpha \Delta s_i$, else if failed ($f$ increased), then $\Delta s_i = -\beta \Delta s_i$.

   until the loop is executed N times

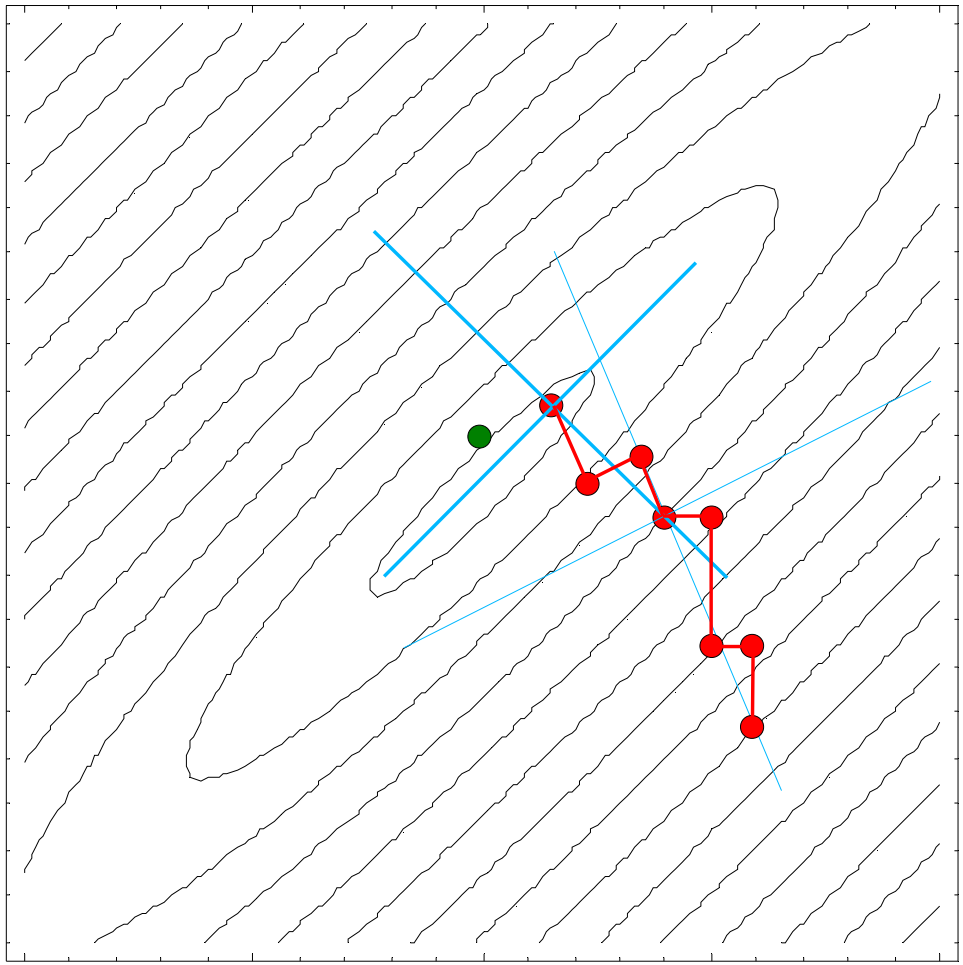
   or until all Δs_i become too small

   or until at least one success and one failure in each direction.

Rosenbrock method requires a cheap objective function.

Zero order methods — Rosenbrock method

Rosenbrock method



Outline

Simplified steepest descent

The steepest descent method is often implemented in the following simplified version:

$$\mathbf{x}_{k+1} = \mathbf{x}_k - \alpha \nabla f(\mathbf{x}_k),$$

where $\alpha > 0$ is a given coefficient. This version

- does not satisfy the Wolfe (strong Wolfe, Goldstein and Price, backtracking) conditions and so
- can perform extremely poorly and should not be used.

However, investigation of its properties reveals astonishing complexity[4]. Consider the following characteristics:

- the number of iterations necessary for the method to converge to the minimum from a given starting point,
- the attraction basins of the minima.

[4] See C. A. Reiter's home page at `http://webbox.lafayette.edu/~reiterc`.

Plotting the number of iterations

The simplified steepest descent method:

$$\mathbf{x}_{k+1} = \mathbf{x}_k - \alpha \nabla f(\mathbf{x}_k).$$

For every value of the coefficient α, the regions of quick and slow convergence of the method can be illustrated using the following characteristics of starting points $\mathbf{x}_0$:

$$n_{\text{steps}}(\mathbf{x}; \alpha, \epsilon) = \arg\min_k \min_m |\mathbf{x}_k - \mathbf{x}_m^\star| < \epsilon, \qquad \text{with } \mathbf{x}_0 = \mathbf{x},$$

where $\mathbf{x}_m^\star$ are all local minima of the objective function f.

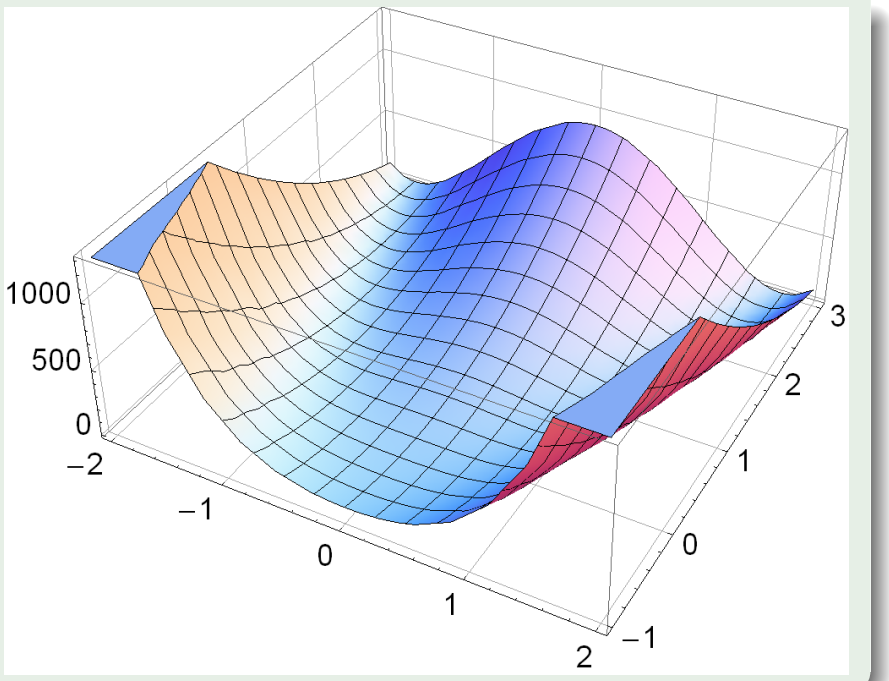
Thus, $n_{\text{steps}}(\mathbf{x}; \alpha, \epsilon)$ is the number of the iterations necessary for $\{\mathbf{x}_k\}$ to converge from $\mathbf{x}$ to any minimum of f.

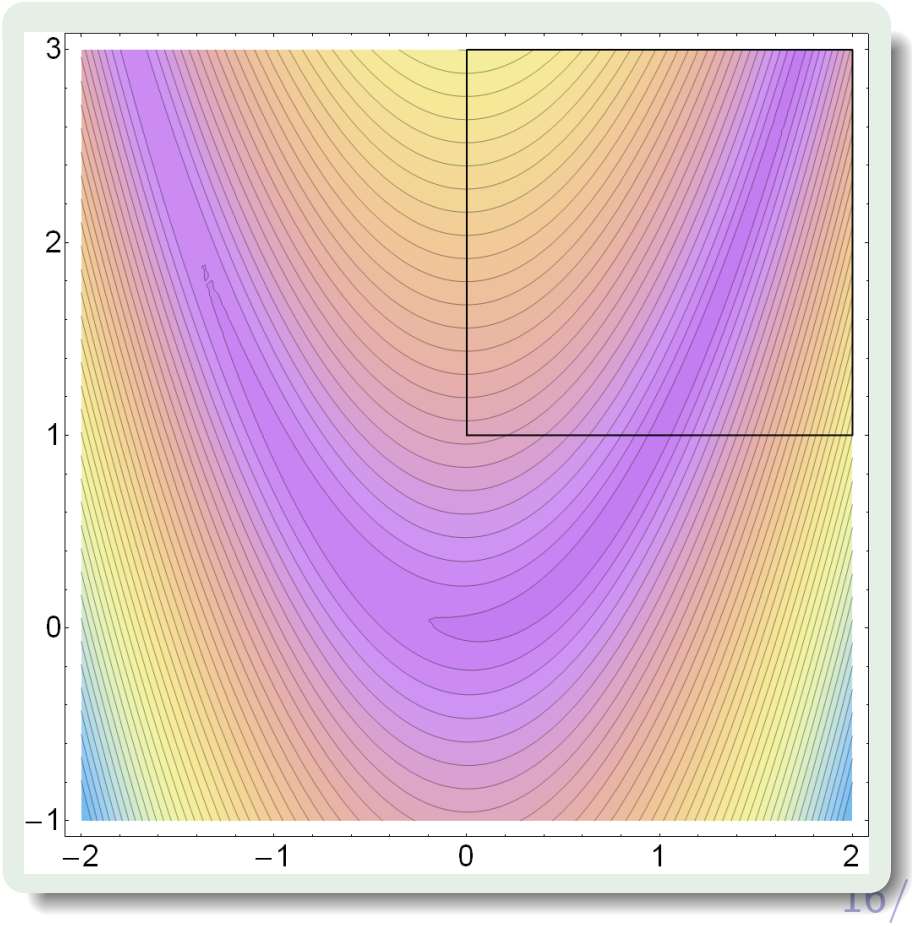
Plotting the number of iterations

Consider the Rosenbrock "banana" function, focus on $[0, 2] \times [1, 3]$

$$f(x, y) = 100(y - x^2)^2 + (x - 1)^2$$

The function f has a single global minimum at $(x^\star, y^\star) = (1, 1)$.





Plotting the number of iterations

Legend

few iterations many iterations

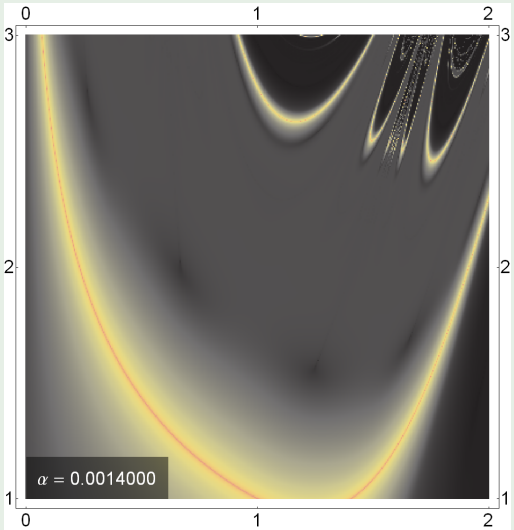
The scale

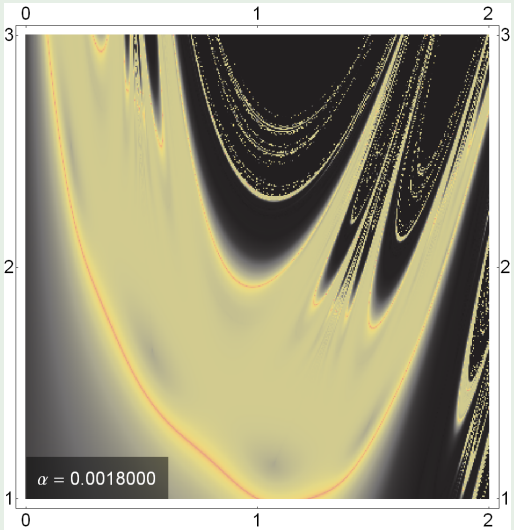
- begins at zero iterations
- ends at the maximum number of iterations in the current frame (but not more than 20 000).

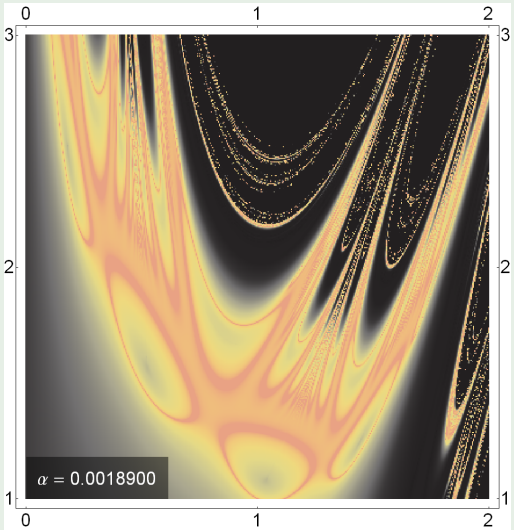
All images have been computed with the resolution 400×400[5]. The accuracy $\epsilon = 0.001$.

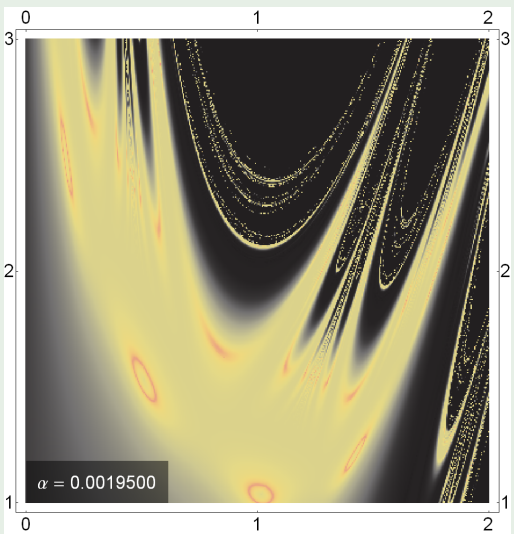
[5] 600×600 for the accompanying video.

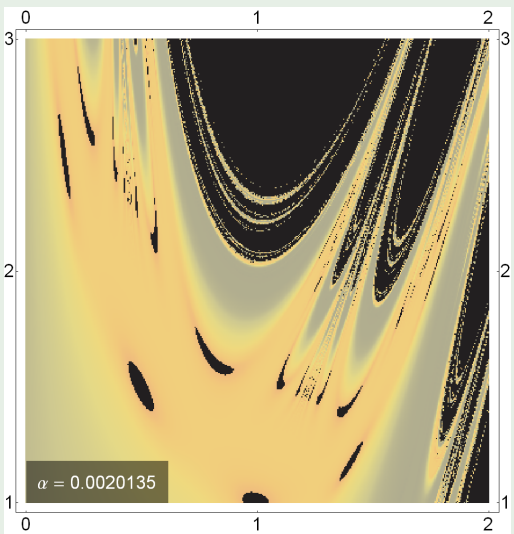
Plotting the number of iterations

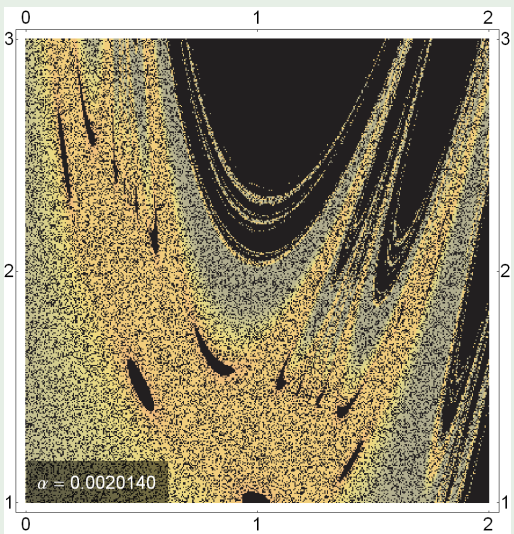




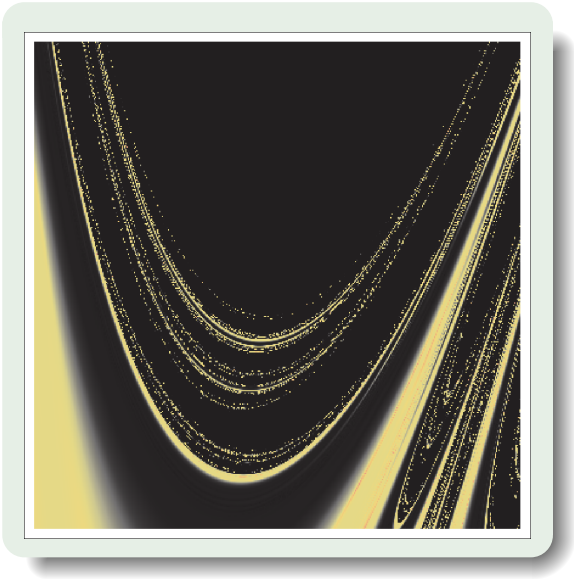
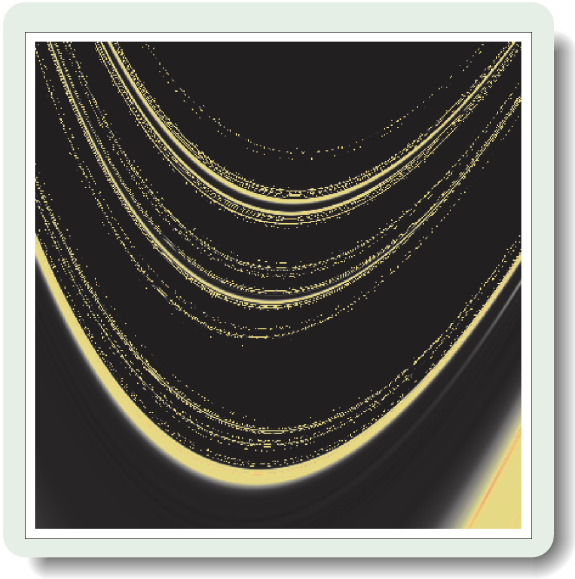
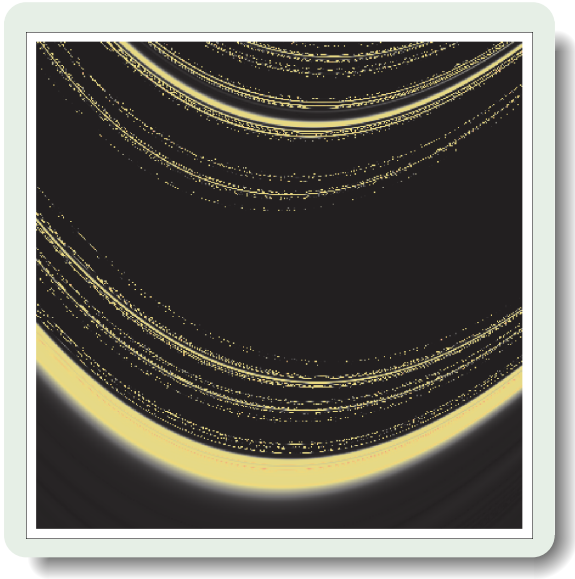
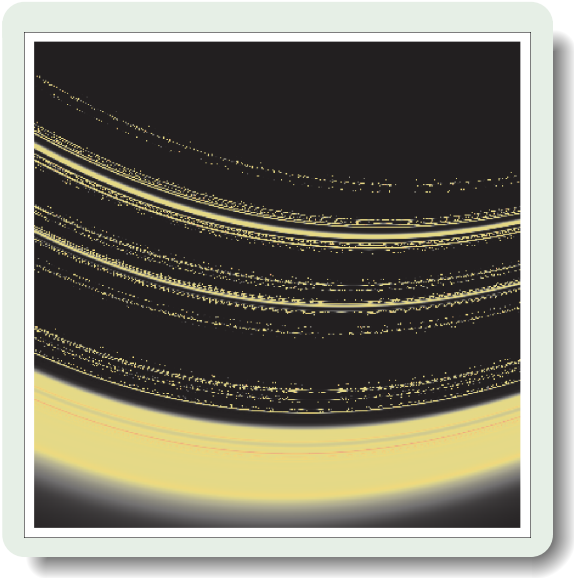
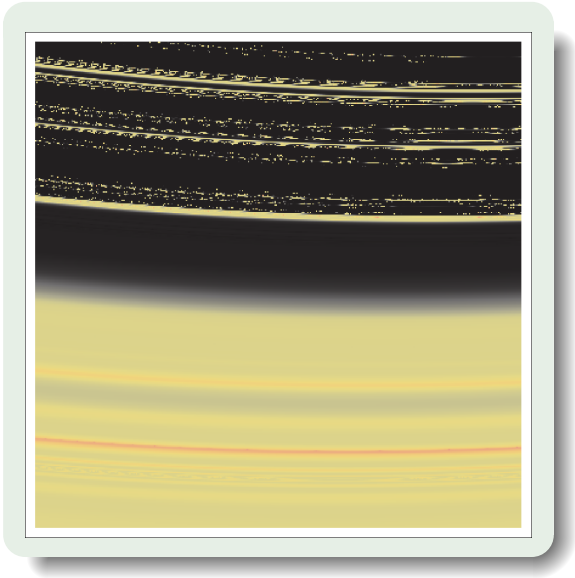
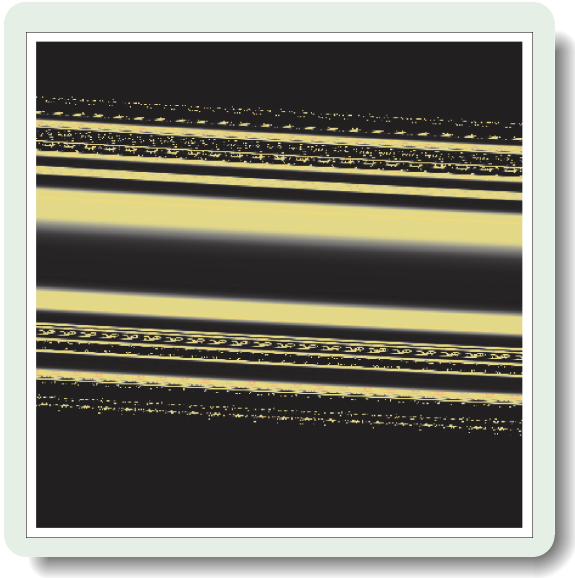








Plotting the number of iterations — zoom for $\alpha = 0.00195$

Plotting the attraction basins of minima

The simplified steepest descent method:

$$\mathbf{x}_{k+1} = \mathbf{x}_k - \alpha \nabla f(\mathbf{x}_k).$$

For every value of the coefficient α, the attraction basins of the minima of the objective function can be illustrated by plotting

$$n_{\min}(\mathbf{x}; \alpha, \epsilon) = \arg\min_m |\mathbf{x}_m^\star - \lim_{k\to\infty} \mathbf{x}_k|, \qquad \mathbf{x}_0 = \mathbf{x},$$

if $\{\mathbf{x}_k\}$ is convergent and 0 otherwise.

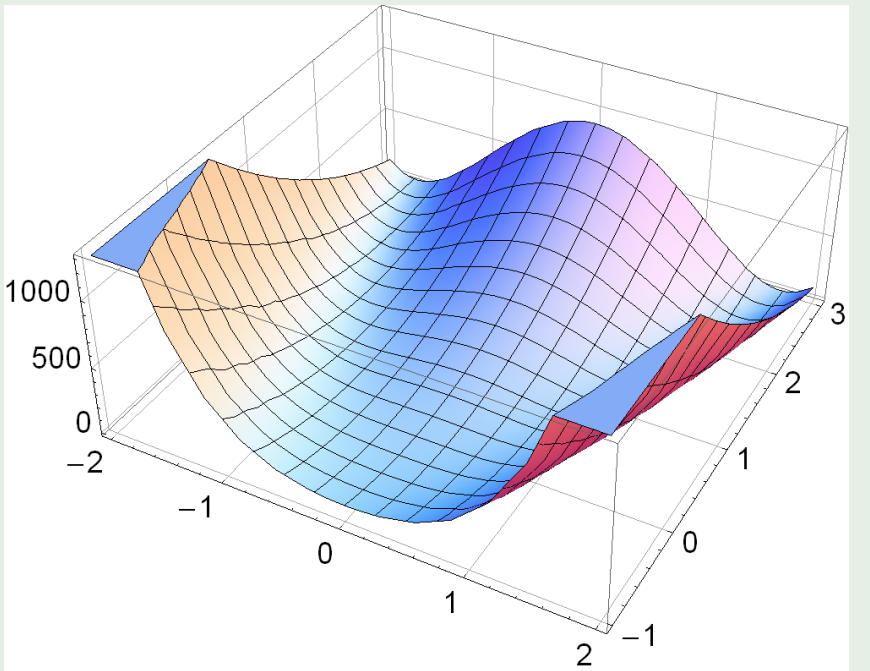
Thus, $n_{\min}(\mathbf{x}; \alpha, \epsilon)$ is the number of the minimum to which $\{\mathbf{x}_k\}$ converges from $\mathbf{x}$ (or 0, if $\{\mathbf{x}_k\}$ is nonconvergent).

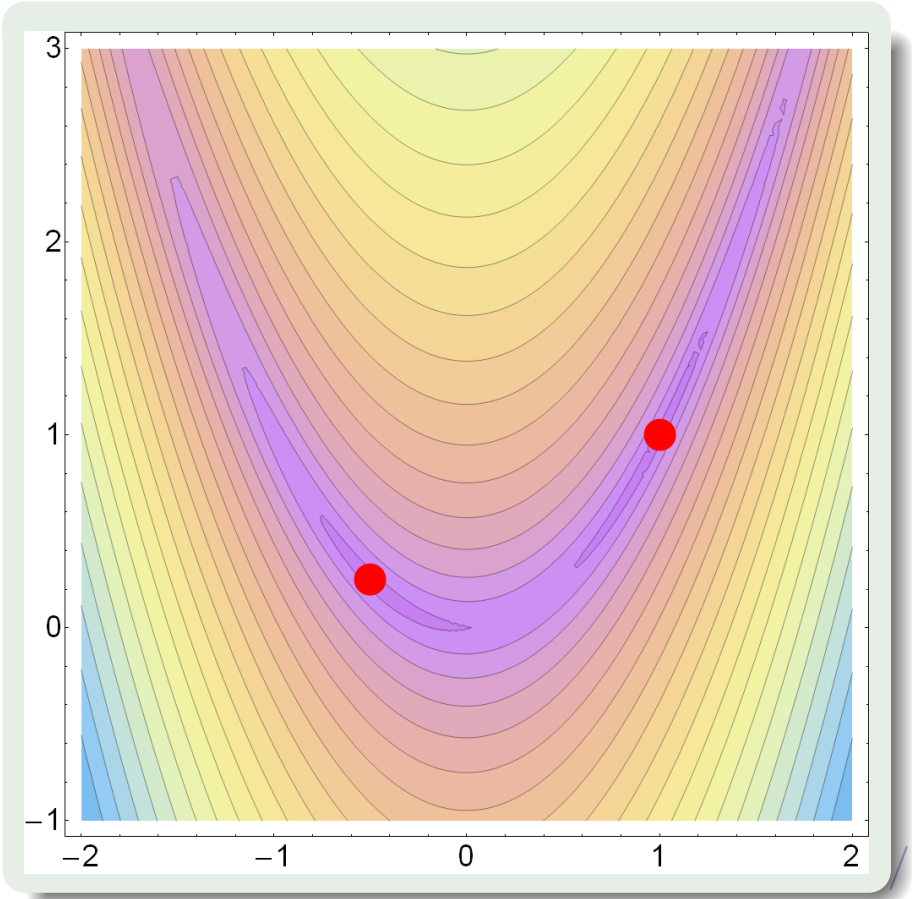
Plotting the attraction basins of minima

Consider a two-minimum modification of the Rosenbrock "banana" function:

$$f(x,y) = 100(y - x^2)^2 + (x + \frac{1}{2})^2(x-1)^2$$

The function f has two global minima at $(-\frac{1}{2}, \frac{1}{4})$ and $(1,1)$.





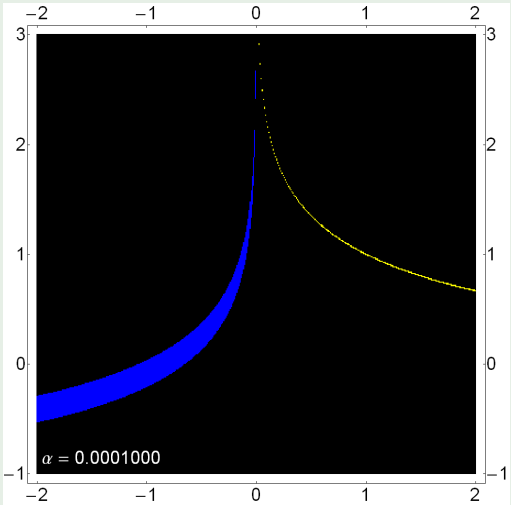
Plotting the attraction basins of minima

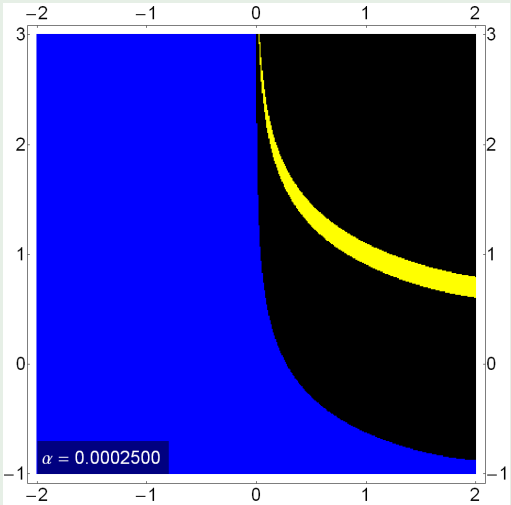
Legend

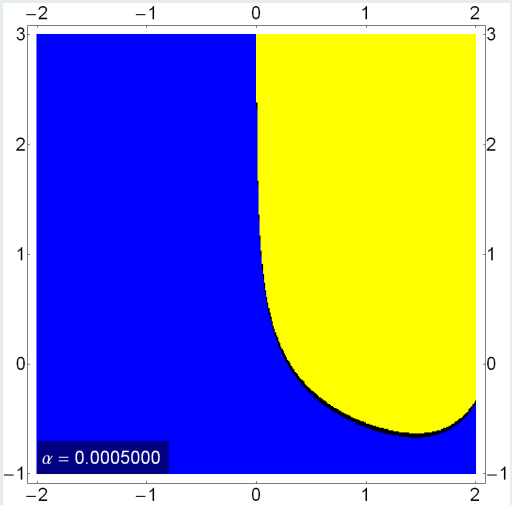
All images have been computed with the resolution 400×400[6].
The accuracy $\epsilon = 0.001$. Up to 20 000 iterations performed.

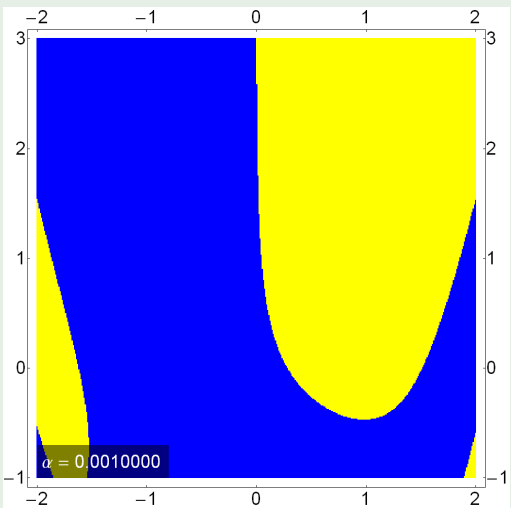
[6] 600×600 for the accompanying video.

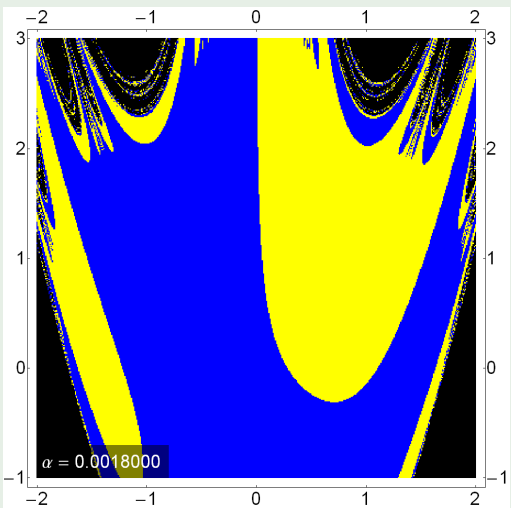
Plotting the attraction basins of minima

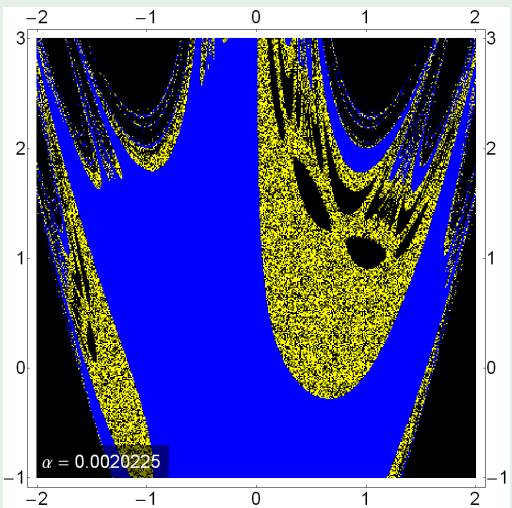












Outline

Search directions

Conjugate direction

Any (smooth enough) function can be approximated by a quadratic form, that is by its second-order Taylor series,

$$f(\mathbf{x}_k + s\,\mathbf{d}_k) \approx f(\mathbf{x}_k) + s\,\mathbf{d}_k^{\mathsf{T}}\nabla f(\mathbf{x}_k) + \frac{1}{2}s^2\mathbf{d}_k^{\mathsf{T}}\nabla^2 f(\mathbf{x}_k)\,\mathbf{d}_k.$$

The gradient of the approximation is

$$\nabla f(\mathbf{x}_k + s\,\mathbf{d}_k) \approx \nabla f(\mathbf{x}_k) + s\,\nabla^2 f(\mathbf{x}_k)\,\mathbf{d}_k.$$

Search directions

Conjugate direction

If $f(\mathbf{x}_k)$ is the exact minimum along the previous search direction $\mathbf{d}_{k-1}$, the gradient at the minimum $\mathbf{x}_k$ is perpendicular to the search direction $\mathbf{d}_{k-1}$,

$$\mathbf{d}_{k-1}^{\mathsf{T}} \nabla f(\mathbf{x}_k) = 0.$$

It is reasonable to expect that the minimization along the next direction $\mathbf{d}_k$ does not jeopardize the minimization along the previous direction $\mathbf{d}_{k-1}$. Therefore, the gradient in the points along the new search direction $\mathbf{d}_k$ should still stay perpendicular to $\mathbf{d}_{k-1}$. Thus we require that

$$\begin{aligned} 0 &= \mathbf{d}_{k-1}^{\mathsf{T}} \nabla f(\mathbf{x}_k + s\,\mathbf{d}_k) \\ &\approx \mathbf{d}_{k-1}^{\mathsf{T}} \left[\nabla f(\mathbf{x}_k) + s\,\nabla^2 f(\mathbf{x}_k)\,\mathbf{d}_k\right] \\ &= s\,\mathbf{d}_{k-1}^{\mathsf{T}} \nabla^2 f(\mathbf{x}_k)\,\mathbf{d}_k. \end{aligned}$$

Conjugate directions

Conjugate direction

Every direction $\mathbf{d}_k$, which satisfies

$$0 = \mathbf{d}_{k-1}^{\mathsf{T}} \nabla^2 f(\mathbf{x}_k)\, \mathbf{d}_k,$$

is said to be conjugate to $\mathbf{d}_{k-1}$ (at $\mathbf{x}$ with respect to f).

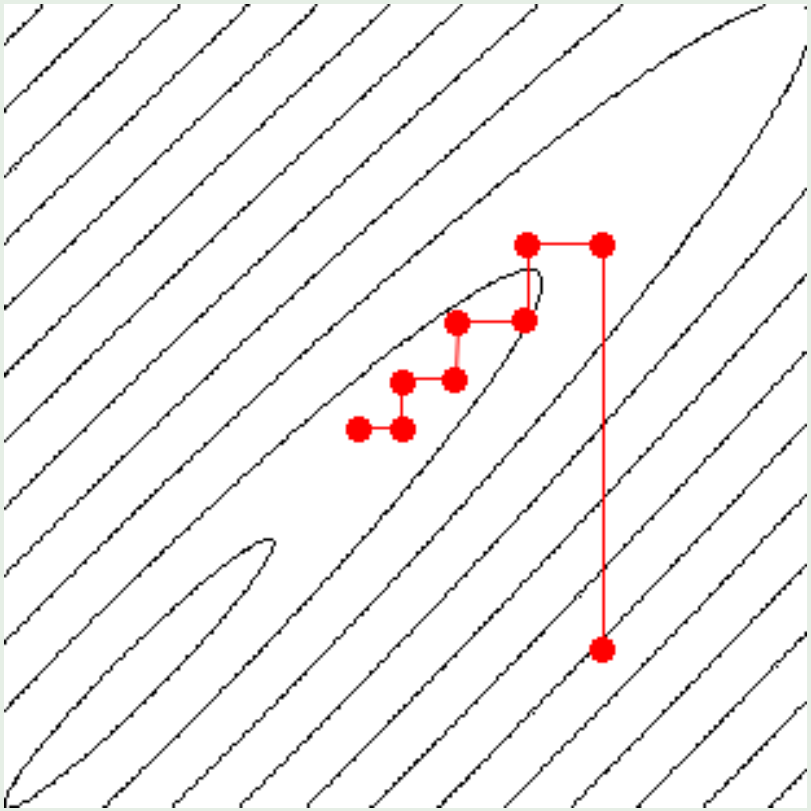
Conjugate set

A set of vectors $\mathbf{d}_i$ that pairwise satisfy $0 = \mathbf{d}_i^{\mathsf{T}} \nabla^2 f(\mathbf{x})\, \mathbf{d}_j$, is called a *conjugate set* (at $\mathbf{x}$ with respect to f).

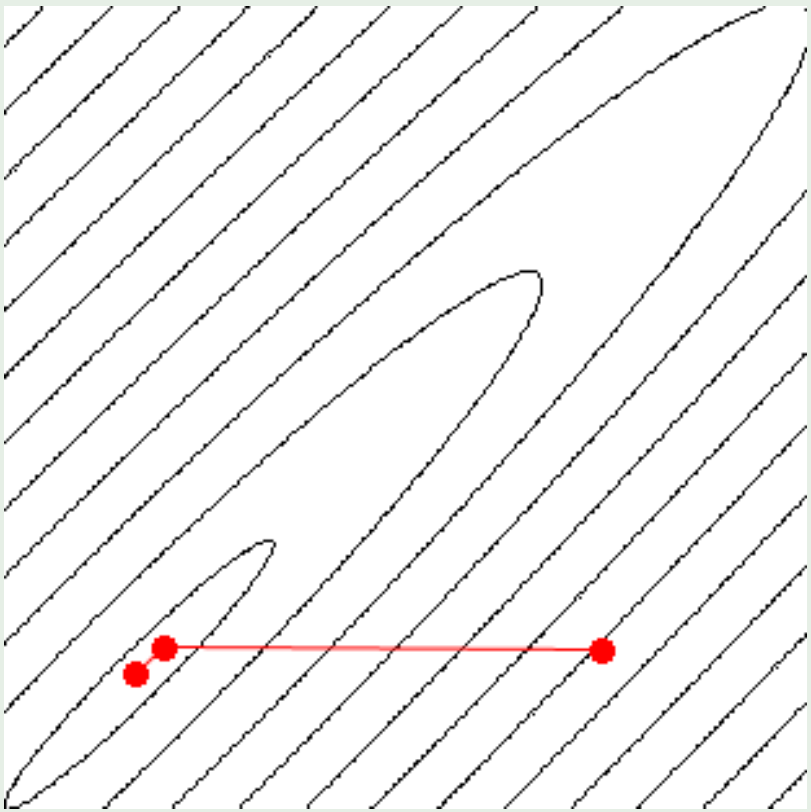
If f is an n-dimensional quadratic form, then n *global* conjugate directions can be always found. If f is also positive definite, then single exact optimization along each of them leads directly to the global minimum.

Conjugate directions

Coordinate descent



Conjugate directions



Linear conjugate directions

Let $\mathbf{A}$ be an $n \times n$ positive definite matrix and

$$f(\mathbf{x}) := c + \mathbf{x}^\mathsf{T}\mathbf{b} + \frac{1}{2}\mathbf{x}^\mathsf{T}\mathbf{A}\mathbf{x},$$

$$\mathbf{r}(\mathbf{x}) := \nabla f(\mathbf{x}) = \mathbf{b} + \mathbf{A}\mathbf{x},$$

and $\mathbf{d}_i$ $(i = 1, 2, \ldots, n)$ be mutually conjugate directions with respect to $\mathbf{A}$,

$$\mathbf{d}_i^\mathsf{T}\mathbf{A}\mathbf{d}_j = 0 \qquad \text{for } i \neq j.$$

Linear conjugate directions

According to the line search principle

$$\mathbf{x}_{k+1} := \mathbf{x}_k + s_k \mathbf{d}_k,$$

where s_k minimizes $f_k(s) := f(\mathbf{x}_k + s\mathbf{d}_k)$. For quadratic f, f_k is a parabola with the exact minimizer

$$s_k = -\frac{\mathbf{d}_k^\mathsf{T} \mathbf{r}_k}{\mathbf{d}_k^\mathsf{T} \mathbf{A} \mathbf{d}_k},$$

where $\mathbf{r}_k = \mathbf{r}(\mathbf{x}_k) = \nabla f(\mathbf{x}_k) = \mathbf{b} + \mathbf{A}\mathbf{x}_k$.
The matrix $\mathbf{A}$ is $n \times n$, hence exactly n such optimum steps along all conjugate directions $\mathbf{d}_i$ lead to the global minimum,

$$\mathbf{x}_n = \mathbf{x}^\star = \mathbf{x}_0 + \sum_{k=0}^{n-1} s_k \mathbf{d}_k.$$

Linear conjugate directions

Choosing the next direction

Let $\mathbf{d}_i$, $i = 0, 1, \ldots, k$ be (already computed) conjugate directions with respect to $\mathbf{A}$. Then

$$\mathbf{x}_{k+1} = \mathbf{x}_0 + \sum_{i=0}^{k} s_i \mathbf{d}_i$$

is the minimum in the subspace

$$\mathbf{x}_0 + \operatorname{span}\{\mathbf{d}_0, \ldots, \mathbf{d}_k\}$$

and thus the gradient $\mathbf{r}_{k+1} = \nabla f(\mathbf{x}_{k+1})$ is perpendicular to $\operatorname{span}\{\mathbf{d}_0, \ldots, \mathbf{d}_k\}$ and so

$$\mathbf{r}_{k+1}^{\mathsf{T}} \mathbf{d}_i = 0, \qquad i = 0, 1, \ldots, k.$$

Linear conjugate directions

Choosing the next direction

The next conjugate direction $\mathbf{d}_{k+1}$ can be hence computed using the gradient $\mathbf{r}_{k+1}$ as

$$\mathbf{d}_{k+1} = -\mathbf{r}_{k+1} + \sum_{i=0}^{k} \eta_{k+1,i}\, \mathbf{d}_i,$$

where the coefficient $\eta_{k+1,i}$ ensure conjugacy of $\mathbf{d}_{k+1}$ with all the previous directions,

$$\mathbf{d}_{k+1}^{\mathsf{T}} \mathbf{A} \mathbf{d}_i = 0, \qquad i = 0, 1, \ldots, k$$

which yields

$$\eta_{k+1,i} = \frac{\mathbf{r}_{k+1}^{\mathsf{T}} \mathbf{A} \mathbf{d}_i}{\mathbf{d}_i^{\mathsf{T}} \mathbf{A} \mathbf{d}_i}.$$

Therefore

$$\mathbf{d}_{k+1} = -\mathbf{r}_{k+1} + \sum_{i=0}^{k} \frac{\mathbf{r}_{k+1}^{\mathsf{T}} \mathbf{A} \mathbf{d}_i}{\mathbf{d}_i^{\mathsf{T}} \mathbf{A} \mathbf{d}_i} \mathbf{d}_i.$$

Linear conjugate directions

Linear conjugate directions (for a quadratic form)

$$\mathbf{d}_{k+1} := -\mathbf{r}_{k+1} + \sum_{i=0}^{k} \eta_{k+1,i}\, \mathbf{d}_i, \qquad \eta_{k+1,i} = \frac{\mathbf{r}_{k+1}^{\mathsf{T}} \mathbf{A}\mathbf{d}_i}{\mathbf{d}_i^{\mathsf{T}} \mathbf{A}\mathbf{d}_i},$$

where

$$f(\mathbf{x}) = c + \mathbf{x}^{\mathsf{T}}\mathbf{b} + \frac{1}{2}\mathbf{x}^{\mathsf{T}}\mathbf{A}\mathbf{x},$$

$$\mathbf{r}(\mathbf{x}) := \nabla f(\mathbf{x}) = \mathbf{b} + \mathbf{A}\mathbf{x}.$$

Therefore, the computation of

- the next conjugate direction $\mathbf{d}_{k+1}$ requires $k+1$ coefficients $\eta_{k+1,i}$ $(i = 0, 1, \ldots, k)$ to be computed
- the entire set of all conjugate directions $\mathbf{d}_i$, $i = 0, \ldots, n-1$,
  - requires $O(n^2)$ time with a large constant and
  - can be numerically unstable (a scheme similar to Gram-Schmidt orthogonalization).

Linear conjugate gradient method

Fortunately, it turns out that if the first direction $\mathbf{d}_0$ is the steepest descent direction,

$$\mathbf{d}_0 = -\mathbf{r}_0 = -\nabla f(\mathbf{x}_0),$$

then in

$$\mathbf{d}_{k+1} := -\mathbf{r}_{k+1} + \sum_{i=0}^{k} \eta_{k+1,i}\, \mathbf{d}_i$$

it is enough to take into account the *last direction* only,

$$\mathbf{d}_{k+1} := -\mathbf{r}_{k+1} + \eta_k\, \mathbf{d}_k = -\mathbf{r}_{k+1} + \frac{\mathbf{r}_{k+1}^{\mathsf{T}} \mathbf{A}\mathbf{d}_k}{\mathbf{d}_k^{\mathsf{T}} \mathbf{A}\mathbf{d}_k} \mathbf{d}_k,$$

and the resulting direction $\mathbf{d}_{k+1}$ is *automatically* conjugate to all previous directions $\mathbf{d}_i$, $i = 0, 1, \ldots, k$.

This is called (a bit misleadingly) the conjugate gradient method.

Linear conjugate gradient method

Using

$$\begin{aligned} \mathbf{r}_{k+1} - \mathbf{r}_k &= s_k \mathbf{A}\mathbf{d}_k, \\ \mathbf{d}_{k+1} &= -\mathbf{r}_{k+1} + \eta_k \mathbf{d}_k, \\ \mathbf{r}_{k+1}^{\mathsf{T}} \mathbf{d}_i &= 0 \qquad \text{for } i = 0, 1, \ldots, k, \end{aligned}$$

it is easy to

- express the optimum step length s_k for the use with $\mathbf{x}_{k+1} := \mathbf{x}_k + s_k \mathbf{d}_k$ as

$$s_k = -\frac{\mathbf{d}_k^{\mathsf{T}} \mathbf{r}_k}{\mathbf{d}_k^{\mathsf{T}} \mathbf{A}\mathbf{d}_k} = \frac{\mathbf{r}_k^{\mathsf{T}} \mathbf{r}_k}{\mathbf{d}_k^{\mathsf{T}} \mathbf{A}\mathbf{d}_k}.$$

- simplify η_k for the use with $\mathbf{d}_{k+1} := -\mathbf{r}_{k+1} + \eta_k \mathbf{d}_k$,

$$\eta_k = \frac{\mathbf{r}_{k+1}^{\mathsf{T}} \mathbf{A}\mathbf{d}_k}{\mathbf{d}_k^{\mathsf{T}} \mathbf{A}\mathbf{d}_k} = \frac{\mathbf{r}_{k+1}^{\mathsf{T}} \mathbf{r}_{k+1}}{\mathbf{r}_k^{\mathsf{T}} \mathbf{r}_k}.$$

Linear conjugate gradient method

Linear conjugate gradient

- Given $f(\mathbf{x}) = c + \mathbf{x}^\mathsf{T}\mathbf{b} + \frac{1}{2}\mathbf{x}^\mathsf{T}\mathbf{A}\mathbf{x}$, chose the initial point $\mathbf{x}_0$.
- Initialize: $\mathbf{r}_0 := \mathbf{b} + \mathbf{A}\mathbf{x}_0$, $\mathbf{d}_0 := -\mathbf{r}_0$ and $k := 0$.

- While stop conditions not satisfied do

$$s_k := \frac{\mathbf{r}_k^\mathsf{T}\mathbf{r}_k}{\mathbf{d}_k^\mathsf{T}\mathbf{A}\mathbf{d}_k},$$
$$\mathbf{x}_{k+1} := \mathbf{x}_k + s_k\mathbf{d}_k,$$
$$\mathbf{r}_{k+1} := \mathbf{r}_k + s_k\mathbf{A}\mathbf{d}_k,$$
$$\eta_k := \frac{\mathbf{r}_{k+1}^\mathsf{T}\mathbf{r}_{k+1}}{\mathbf{r}_k^\mathsf{T}\mathbf{r}_k},$$
$$\mathbf{d}_{k+1} := -\mathbf{r}_{k+1} + \eta_k\,\mathbf{d}_k,$$
$$k := k + 1.$$

This is basically the CG method for solving $\mathbf{A}\mathbf{x} = -\mathbf{b}$ (Lecture B-3).

Nonlinear conjugate gradient method

The conjugate gradient algorithm can be (almost) directly used with non-quadratic objective functions, provided the exactly minimizing step is replaced with a line search. It is inexpensive numerically (no Hessian, just gradients; data storage only one step back) and in general yields a superlinear convergence.

The line search need not to be exact. However, the step length s_k has to satisfy the strong Wolfe conditions with $0 < c_1 < c_2 < 0.5$ (Lecture C-2) in order to assure that the next direction $\mathbf{d}_{k+1} := -\mathbf{r}_{k+1} + \eta_k \mathbf{d}_k$ is a descent direction with

$$\eta_k := \frac{\mathbf{r}_{k+1}^{\mathsf{T}} \mathbf{r}_{k+1}}{\mathbf{r}_k^{\mathsf{T}} \mathbf{r}_k} = \frac{\nabla f(\mathbf{x}_{k+1})^{\mathsf{T}} \nabla f(\mathbf{x}_{k+1})}{\nabla f(\mathbf{x}_k)^{\mathsf{T}} \nabla f(\mathbf{x}_k)}.$$

Nonlinear conjugate gradient method

Nonlinear conjugate gradient

- Given $f(\mathbf{x})$, chose the initial point $\mathbf{x}_0$.
- Initialize: $\mathbf{r}_0 := \nabla f(\mathbf{x}_0)$, $\mathbf{d}_0 := -\mathbf{r}_0$ and $k := 0$.
- While stop conditions not satisfied do
  1. Find step length s_k satisfying strong Wolfe conditions with $0 < c_1 < c_2 < 0.5$, set $\mathbf{x}_{k+1} := \mathbf{x}_k + s_k \mathbf{d}_k$.
  2. Compute $\mathbf{r}_{k+1} := \nabla f(\mathbf{x}_{k+1})$.
  3. Compute

Fletcher-Reeves

$$\eta_k := \frac{\mathbf{r}_{k+1}^\mathsf{T} \mathbf{r}_{k+1}}{\mathbf{r}_k^\mathsf{T} \mathbf{r}_k}$$

Polak-Ribière

$$\eta_k := \max\left[0, \frac{\mathbf{r}_{k+1}^\mathsf{T} (\mathbf{r}_{k+1} - \mathbf{r}_k)}{\mathbf{r}_k^\mathsf{T} \mathbf{r}_k}\right]$$

  4. Compute $\mathbf{d}_{k+1} := -\mathbf{r}_{k+1} + \eta_k \mathbf{d}_k$.
  5. Update the step number $k := k + 1$.

Nonlinear conjugate gradient method

Iteratively generated search directions $\mathbf{d}_k$ can tend to "fold up on each other" and become linearly dependent. Therefore, the directions should be periodically reset by assuming $\eta_k := 0$, which effectively restarts the procedure of generating the directions from the scratch (that is, the steepest descent direction).

- Restart every N iterations.
- With quadratic objective function and exact line searches, consecutive gradients are orthogonal, $\mathbf{r}_i^\mathsf{T}\mathbf{r}_j = 0$ if $i \neq j$. The procedure can be thus restarted when

$$\frac{\mathbf{r}_{k+1}^\mathsf{T}\mathbf{r}_k}{\|\mathbf{r}_{k+1}\|\|\mathbf{r}_k\|} > c \approx 0.1,$$

  which in practice happens rather frequently.
- Polak-Ribière method restarts anyway when the computed correction term is negative (rather infrequently).

Outline

Newton methods

The Newton direction $\mathbf{d}_k := -\left[\nabla^2 f(\mathbf{x}_k)\right]^{-1} \nabla f(\mathbf{x}_k)$ is based on the quadratic approximation to the objective function:

$$f(\mathbf{x}_k + \mathbf{x}) \approx f(\mathbf{x}_k) + \mathbf{x}^{\mathsf{T}} \nabla f(\mathbf{x}_k) + \frac{1}{2} \mathbf{x}^{\mathsf{T}} \nabla^2 f(\mathbf{x}_k)\, \mathbf{x}.$$

If the Hessian $\nabla^2 f(\mathbf{x}_k)$ is positive definite (the approximation is convex), the minimum is found by solving

$$\nabla f(\mathbf{x}_k + \mathbf{d}_k) \approx \nabla f(\mathbf{x}_k) + \nabla^2 f(\mathbf{x}_k)\, \mathbf{d}_k = \mathbf{0}.$$

- The Hessian has to be computed (a 2^{nd} order method).
- Inverting a large Hessian is time-consuming.
- Far from the minimum the Hessian may not be positive definite and $\mathbf{d}_k$ may be an ascent direction.

- Quick quadratic convergence near the minimum.

Newton methods

Far from the minimum

- the Hessian may not be positive definite and
- the exact Newton direction may be an ascent direction.

In practice the Newton method is implemented as

Inexact Newton methods, which solve $\nabla^2 f(\mathbf{x}_k)\,\mathbf{d}_k = -\nabla f(\mathbf{x}_k)$ inexactly to assure that $\mathbf{d}_k$ is a descent direction (Newton–conjugate gradient method).

Modified Newton methods, which modify the Hessian matrix $\nabla^2 f(\mathbf{x}_n)$ so that it becomes positive definite. The solution to $\mathbf{H}_k\,\mathbf{d}_k = -\nabla f(\mathbf{x}_k)$, where $\mathbf{H}_k$ is the modified Hessian, is then a descent direction.

Inexact Newton methods

Inexact solution of $\nabla^2 f(\mathbf{x}_k)\,\mathbf{d}_k = -\nabla f(\mathbf{x}_k)$

- Is quicker, since the Hessian is not inverted.
- Can guarantee that the inexact solution is a descent direction.

Stop condition is usually based on the norm of the residuum, normalized with respect to the norm of the RHS ($\nabla f(\mathbf{x}_k)$):

$$\frac{\|\mathbf{r}_k\|}{\|\nabla f(\mathbf{x}_k)\|} = \frac{\|\nabla^2 f(\mathbf{x}_k)\,\mathbf{d}_k + \nabla f(\mathbf{x}_k)\|}{\|\nabla f(\mathbf{x}_k)\|} \leq \alpha_k,$$

where at least $\alpha_k < \alpha < 1$ or preferably $\alpha_k \to 0$, for example

$$\alpha_k = \min\left[\frac{1}{2}, \|\nabla f(\mathbf{x}_k)\|\right] \qquad \text{or} \qquad \alpha_k = \min\left[\frac{1}{2}, \sqrt{\|\nabla f(\mathbf{x}_k)\|}\right].$$

Inexact Newton methods

Newton–conjugate gradient

The Newton–conjugate gradient method solves in each step

$$\nabla^2 f(\mathbf{x}_k)\,\mathbf{d}_k \;=\; -\nabla f(\mathbf{x}_k)$$

iteratively (an iteration in each step of the iteration) using the linear conjugate gradient method, with the stop conditions

- Direction $\mathbf{p}_{i+1}$ of a negative curvature is generated,

$$\mathbf{p}_{i+1}^{\mathsf{T}} \nabla^2 f(\mathbf{x}_k) \mathbf{p}_{i+1} \leq 0,$$

- or the Newton equation is inexactly solved,

$$\frac{\|\mathbf{r}_k\|}{\|\nabla f(\mathbf{x}_k)\|} \leq \alpha_k,$$

where

$$\alpha_k = \min\left[\frac{1}{2}, \sqrt{\|\nabla f(\mathbf{x}_k)\|}\right].$$

Modified Newton methods

Modified Newton methods modify the Hessian matrix $\nabla^2 f(\mathbf{x}_k)$ (as little as possible) by

$$\mathbf{H}_k = \nabla^2 f(\mathbf{x}_k) + \mathbf{E}_k,$$

so that it becomes *sufficiently positive definite* and the solution to the modified equation

$$\left[\nabla^2 f(\mathbf{x}_k) + \mathbf{E}_k\right] \mathbf{d}_k = -\nabla f(\mathbf{x}_k)$$

is a descent direction. There are several possibilities to choose $\mathbf{E}_k$:

1. A multiple of identity, $\mathbf{E}_k = \tau \mathbf{I}$.
2. Obtained during Cholesky factorisation of the Hessian with immediate increase of the diagonal elements (Cholesky modification).
3. others (Gershgorin modification, etc.).

Outline

Quasi-Newton methods

All Newton methods require the Hessian to be computed and some of them (modified Newton) invert or factorize it. Both tasks are time-consuming and error-prone.

Quasi-Newton methods compute the search direction using a gradient-based approximation to the Hessian (or to its inverse):

$$\mathbf{d}_k = -\mathbf{H}_k^{-1}\nabla f(\mathbf{x}_k) \qquad \text{or} \qquad \mathbf{d}_k = -\mathbf{B}_k\nabla f(\mathbf{x}_k).$$

Quasi-Newton methods are useful

- in large problems (the Hessian is dense and too large),
- with severely ill-conditioned Hessians,
- when second derivatives are unavailable.

Quasi-Newton methods

Approximating the Hessian

Approximation is based on gradients and updated after each step using the previous step length and the change of the gradient.

Approximate the function around the last two iterates:

$$f_{k-1}(\mathbf{x}) := f(\mathbf{x}_{k-1} + \mathbf{x}) \approx f(\mathbf{x}_{k-1}) + \mathbf{x}^\mathsf{T} \nabla f(\mathbf{x}_{k-1}) + \frac{1}{2}\mathbf{x}^\mathsf{T}\mathbf{H}_{k-1}\mathbf{x},$$

$$f_k(\mathbf{x}) := f(\mathbf{x}_k + \mathbf{x}) \approx f(\mathbf{x}_k) + \mathbf{x}^\mathsf{T} \nabla f(\mathbf{x}_k) + \frac{1}{2}\mathbf{x}^\mathsf{T}\mathbf{H}_k\mathbf{x},$$

where $\mathbf{x}_k := \mathbf{x}_{k-1} + s_{k-1}\mathbf{d}_{k-1}$.

Assume the previous-step approximate Hessian $\mathbf{H}_{k-1}$ is known. The next approximation $\mathbf{H}_k$ can be obtained by comparing the gradients in $\mathbf{x}_{k-1}$:

$$\nabla f_k(-s_{k-1}\mathbf{d}_{k-1}) = \nabla f_{k-1}(\mathbf{0}).$$

Quasi-Newton methods

Approximating the Hessian

The requirement $\nabla f_k(-s_{k-1}\mathbf{d}_{k-1}) = \nabla f_{k-1}(\mathbf{0})$ leads to

$$\mathbf{H}_k(\mathbf{x}_k - \mathbf{x}_{k-1}) = \nabla f(\mathbf{x}_k) - \nabla f(\mathbf{x}_{k-1}),$$

which is usually stated in the shorter form and called the

secant equation

$$\mathbf{H}_k\mathbf{y}_k = \mathbf{g}_k,$$

where $\mathbf{y}_k := \mathbf{x}_k - \mathbf{x}_{k-1}$,

$\mathbf{g}_k := \nabla f(\mathbf{x}_k) - \nabla f(\mathbf{x}_{k-1})$.

Such symmetric positive definite $\mathbf{H}_k$ exists (non-uniquely) only if

$$\mathbf{y}_l^T\mathbf{H}_k\mathbf{y}_k = \mathbf{y}_k^T\mathbf{g}_k > 0,$$

which is guaranteed if the step length s_k satisfies Wolfe conditions.

Quasi-Newton methods

Approximating the Hessian

Since the solution is non-unique, $\mathbf{H}_k$ should be chosen so that it is the closest approximate to the last step approximation $\mathbf{H}_{k-1}$:

> *Find a symmetric positive definite matrix* $\mathbf{H}_k$*, which satisfies the secant equation* $\mathbf{H}_k\mathbf{y}_k = \mathbf{g}_k$ *and minimizes* $\|\mathbf{H}_k - \mathbf{H}_{k-1}\|$ *for a given norm* $\|\cdot\|$*.*

The approximation is used to find the quasi-Newton direction $\mathbf{d}_k = -\mathbf{H}_k^{-1}\nabla f(\mathbf{x}_k)$, which requires inverting the approximated Hessian. The task is thus often reformulated to find an approximation to the inverse $\mathbf{B}_k = \mathbf{H}_k^{-1}$:

> *Find a symmetric positive definite matrix* $\mathbf{B}_k$*, which satisfies the inverse secant equation* $\mathbf{B}_k\mathbf{g}_k = \mathbf{y}_k$ *and minimizes* $\|\mathbf{B}_k - \mathbf{B}_{k-1}\|$ *for a given norm* $\|\cdot\|$*.*

Quasi-Newton methods

Approximating the Hessian

In both cases, the solution is easy to find if the weighted Frobenius norm[7] is used,

$$\|\mathbf{A}\|_{\mathbf{W}} = \|\mathbf{W}^{1/2}\mathbf{A}\mathbf{W}^{1/2}\|_{\mathrm{F}},$$

where the the weighting matrix $\mathbf{W}$ satisfies the inverse (or direct) secant equation. If $\mathbf{W}^{-1}$ (or $\mathbf{W}$) is the average Hessian over the last step,

$$\int_0^1 \nabla^2 f(\mathbf{x}_{k-1} + \tau s_{k-1}\mathbf{d}_{k-1})\, d\tau,$$

then two updating formulas are obtained:

DFP (Davidon, Fletcher, Powell) formula for Hessian updating and

BFGS (Broyden, Fletcher, Goldfarb, Shanno) formula for inverse Hessian updating.

Both formulas use rank two updates to the previous step matrix.

[7] The Frobenius norm of a matrix $\mathbf{A}$ is defined as $\|\mathbf{A}\|_{\mathrm{F}}^2 = \sum_{i,j} a_{ij}^2$.

Quasi-Newton methods — DFP method

DFP (Davidon, Fletcher, Powell) Hessian updating formula

$$\mathbf{H}_k^{\mathrm{DFP}} = \left(\mathbf{I} - \frac{\mathbf{g}_k \mathbf{y}_k^{\mathsf{T}}}{\mathbf{g}_k^{\mathsf{T}} \mathbf{y}_k}\right) \mathbf{H}_{k-1} \left(\mathbf{I} - \frac{\mathbf{y}_k \mathbf{g}_k^{\mathsf{T}}}{\mathbf{g}_k^{\mathsf{T}} \mathbf{y}_k}\right) + \frac{\mathbf{g}_k \mathbf{g}_k^{\mathsf{T}}}{\mathbf{g}_k^{\mathsf{T}} \mathbf{y}_k}$$

The corresponding formula for updating the inverse Hessian

$$\mathbf{B}_k^{\mathrm{DFP}} = \mathbf{B}_{k-1} - \frac{\mathbf{B}_{k-1} \mathbf{g}_k \mathbf{g}_k^{\mathsf{T}} \mathbf{B}_{k-1}}{\mathbf{g}_k^{\mathsf{T}} \mathbf{B}_{k-1} \mathbf{g}_k} + \frac{\mathbf{y}_k \mathbf{y}_k^{\mathsf{T}}}{\mathbf{g}_k^{\mathsf{T}} \mathbf{y}_k}$$

Quasi-Newton methods — BFGS method

BFGS (Broyden, Fletcher, Goldfarb, Shanno) inverse Hessian updating formula

$$\mathbf{B}_k^{\text{BFGS}} = \left(\mathbf{I} - \frac{\mathbf{y}_k \mathbf{g}_k^{\mathsf{T}}}{\mathbf{g}_k^{\mathsf{T}} \mathbf{y}_k}\right) \mathbf{B}_{k-1} \left(\mathbf{I} - \frac{\mathbf{g}_k \mathbf{y}_k^{\mathsf{T}}}{\mathbf{g}_k^{\mathsf{T}} \mathbf{y}_k}\right) + \frac{\mathbf{y}_k \mathbf{y}_k^{\mathsf{T}}}{\mathbf{g}_k^{\mathsf{T}} \mathbf{y}_k}$$

The corresponding formula for updating the Hessian

$$\mathbf{H}_k^{\text{BFGS}} = \mathbf{H}_{k-1} - \frac{\mathbf{H}_{k-1} \mathbf{y}_k \mathbf{y}_k^{\mathsf{T}} \mathbf{H}_{k-1}}{\mathbf{y}_k^{\mathsf{T}} \mathbf{H}_{k-1} \mathbf{y}_k} + \frac{\mathbf{g}_k \mathbf{g}_k^{\mathsf{T}}}{\mathbf{g}_k^{\mathsf{T}} \mathbf{y}_k}$$

Quasi-Newton methods — Broyden class

The DFP and the BFGS formulas for updating the Hessian can be linearly combined to form a general updating formula for the Broyden class methods.

Broyden class methods

$$\mathbf{H}_k = (1-\phi)\mathbf{H}_k^{\text{BFGS}} + \phi\,\mathbf{H}_k^{\text{DFP}}.$$

The *restricted Broyden class* is defined by $0 \leq \phi \leq 1$.

Quasi-Newton methods — SR1 method

A member of the Broyden class is the SR1 (symmetric rank one) method, which yields rank one updates to the Hessian.

SR1 method

$$\phi_k^{\mathrm{SR1}} = \frac{\mathbf{y}_k^{\mathsf{T}} \mathbf{g}_k}{\mathbf{y}_k^{\mathsf{T}} \left(\mathbf{g}_k - \mathbf{H}_{k-1} \mathbf{y}_k \right)}$$

The weighting coefficient ϕ_k^{SR1} may fall outside $[0, 1]$ interval.

The SR1 formula approximates Hessian well, but

- the approximated Hessian might not be positive semidefinite,
- the denominator in the above formula can vanish or be small.

It is thus usually used with modified Newton methods (trust region) and sometimes the update is skipped.

Outline

7. Least-squares problems
   - Hessian approximation
   - Gauss-Newton method
   - Levenberg-Marquardt method

Least-squares problems

In many problems (like parameter fitting, structural optimization, etc.) the objective function is a sum of squares (residuals):

$$f(\mathbf{x}) = \frac{1}{2}\mathbf{r}(\mathbf{x})^{\mathsf{T}}\mathbf{r}(\mathbf{x}) = \frac{1}{2}\sum_{i=1}^{n} r_i^2(\mathbf{x}).$$

Then

$$\begin{aligned}
\nabla f(\mathbf{x}) &= \sum_{i=1}^{n} r_i(\mathbf{x})\nabla r_i(\mathbf{x}) = \mathbf{J}(\mathbf{x})^{\mathsf{T}}\mathbf{r}(\mathbf{x}), \\
\nabla^2 f(\mathbf{x}) &= \sum_{i=1}^{n} \nabla r_i(\mathbf{x})\nabla r_i(\mathbf{x})^{\mathsf{T}} + \sum_{i=1}^{n} r_i(\mathbf{x})\nabla^2 r_i(\mathbf{x}) \\
&= \mathbf{J}(\mathbf{x})^{\mathsf{T}}\mathbf{J}(\mathbf{x}) + \sum_{i=1}^{n} r_i(\mathbf{x})\nabla^2 r_i(\mathbf{x}),
\end{aligned}$$

where $\mathbf{J}(\mathbf{x})$ is the Jacobi matrix of the residuals, $\mathbf{J}(\mathbf{x}) = \left[\frac{\partial r_i(\mathbf{x})}{\partial x_j}\right]_{i,j}$.

Least-squares problems — Hessian approximation

If the residuals $\mathbf{r}_i(\mathbf{x})$, at least near the minimum,

- vanish (the optimum point yields is a good fit between the model and the data), that is $r_i(\mathbf{x}) \approx 0$,
- or are linear functions of $\mathbf{x}$, that is $\nabla^2 r_i(\mathbf{x}) \approx \mathbf{0}$,

then the Hessian simplifies to

$$\begin{aligned}\nabla^2 f(\mathbf{x}) &= \mathbf{J}(\mathbf{x})^{\mathsf{T}} \mathbf{J}(\mathbf{x}) + \sum_{i=1}^{n} r_i(\mathbf{x}) \nabla^2 r_i(\mathbf{x}) \\ &\approx \mathbf{J}(\mathbf{x})^{\mathsf{T}} \mathbf{J}(\mathbf{x}),\end{aligned}$$

which is always positive semidefinite and can be computed using the first derivatives of the residuals only.

Least-squares problems — Gauss-Newton method

Substitution of the approximation into the Newton formula,

$$\nabla^2 f(\mathbf{x}_k)\,\mathbf{d}_k = -\nabla f(\mathbf{x}_k),$$

yields the

Gauss-Newton formula

$$\mathbf{J}(\mathbf{x}_k)^{\mathsf{T}}\mathbf{J}(\mathbf{x}_k)\,\mathbf{d}_k = -\mathbf{J}(\mathbf{x}_k)^{\mathsf{T}}\mathbf{r}(\mathbf{x}_k),$$

which may be problematic, if $\mathbf{J}(\mathbf{x}_k)^{\mathsf{T}}\mathbf{J}(\mathbf{x}_k)$ is not a good approximation to the Hessian.

Least-squares problems — Levenberg-Marquardt method

A modified version of the Gauss-Newton method is called the Levenberg-Marquardt method[8]

Levenberg-Marquardt formula

$$\left[\mathbf{J}(\mathbf{x}_k)^\mathsf{T}\mathbf{J}(\mathbf{x}_k) + \lambda\mathbf{I}\right]\mathbf{d}_k = -\mathbf{J}(\mathbf{x}_k)^\mathsf{T}\mathbf{r}(\mathbf{x}_k).$$

The parameter λ allows the step length to be smoothly controlled:

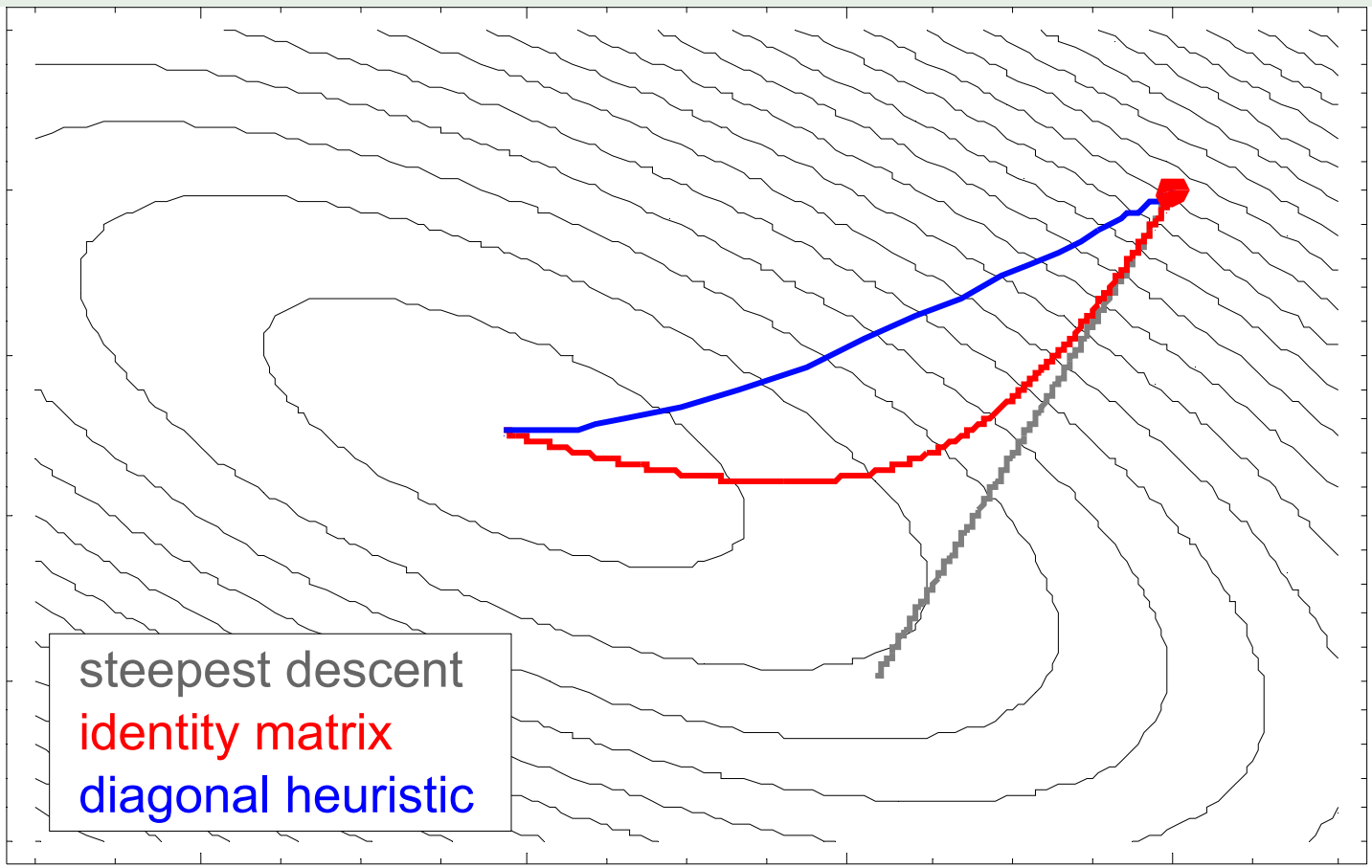
- For small λ, the steps are Gauss-Newton (or Newton) in character and take advantage of the superlinear convergence near the minimum.
- For large λ, the identity matrix dominates and the steps are similar to steepest descent steps.

Sometimes a heuristic is advocated, which uses the diagonal of the approximated Hessian $\mathbf{J}(\mathbf{x}_k)^\mathsf{T}\mathbf{J}(\mathbf{x}_k)$ instead of the identity matrix.

[8] Notice that this is essentially a trust-region approach.

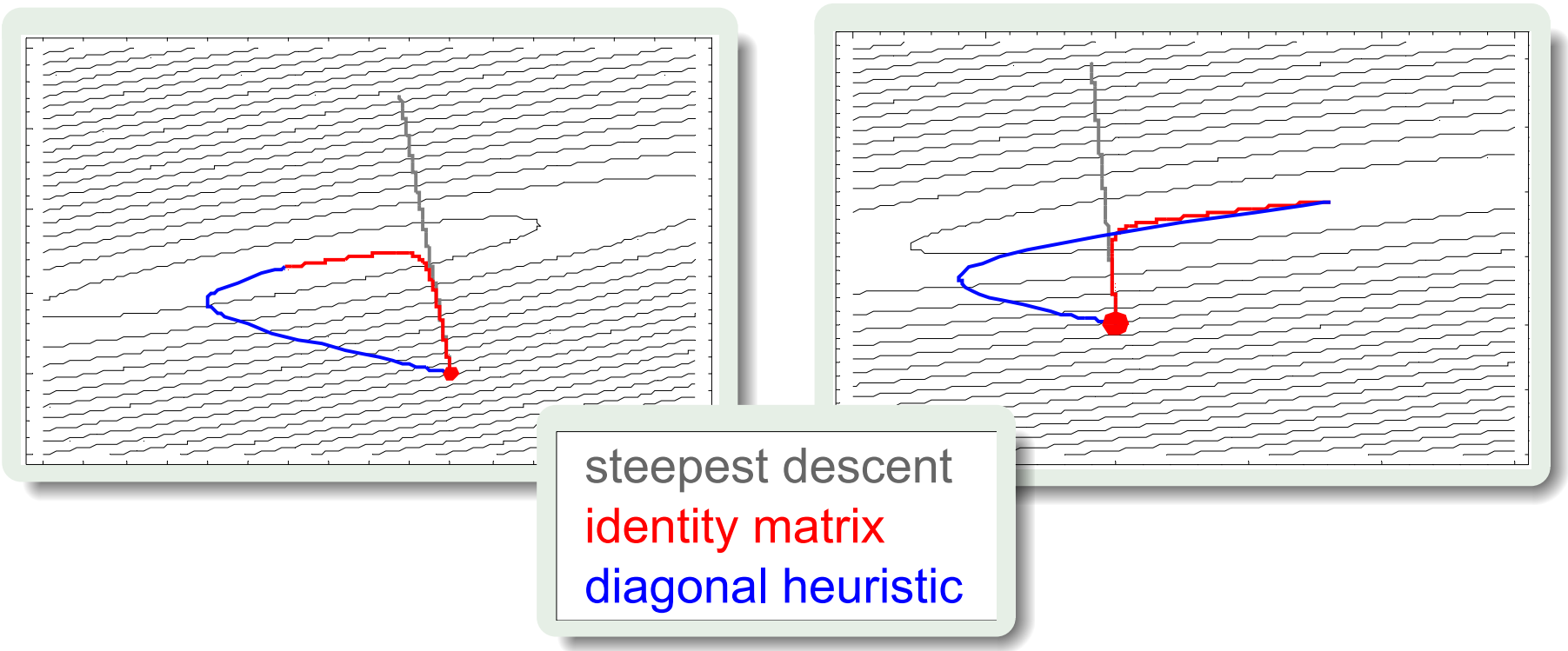
Least-squares problems — Levenberg-Marquardt method

(quadratically) approximated objective function



Least-squares problems — Levenberg-Marquardt method

The rule with the diagonal of the Hessian matrix (instead of the identity) is a heuristic only. It can be quicker, but sometimes (and not so rare) may be also much slower.



Least-squares problems — Levenberg-Marquardt method

In the Levenberg-Marquardt formula,

$$\left[\mathbf{J}(\mathbf{x}_k)^{\mathsf{T}}\mathbf{J}(\mathbf{x}_k) + \lambda \mathbf{I}\right] \mathbf{d}_k = -\mathbf{J}(\mathbf{x}_k)^{\mathsf{T}}\mathbf{r}(\mathbf{x}_k),$$

the parameter λ controls the step length. It is updated in each optimization step based on the accuracy

$$\rho = \frac{f(\mathbf{x}_k) - f(\mathbf{x}_k + \mathbf{d}_k)}{f_k(\mathbf{0}) - f_k(\mathbf{d}_k)} = \frac{\text{actual decrease}}{\text{predicted decrease}}$$

of the quadratic approximation f_k to the objective function

$$f(\mathbf{x}_k + \mathbf{x}) \approx f_k(\mathbf{x}) := f(\mathbf{x}_k) + \mathbf{x}^{\mathsf{T}}\mathbf{J}(\mathbf{x}_k)^{\mathsf{T}}\mathbf{r}(\mathbf{x}_k) + \frac{1}{2}\mathbf{x}^{\mathsf{T}}\mathbf{J}(\mathbf{x}_k)^{\mathsf{T}}\mathbf{J}(\mathbf{x}_k)\mathbf{x}.$$

The denominator in the formula for ρ simplifies to

$$f_k(\mathbf{0}) - f_k(\mathbf{d}_k) = \frac{1}{2}\mathbf{d}_k \left[\lambda\, \mathbf{d}_k - \nabla f(\mathbf{x}_k)\right].$$

Least-squares problems — Levenberg-Marquardt method

Given the coefficients $\rho_{min} \approx 0.1$, $\rho_{\max} \approx 0.75$ and $k \approx 25$.

Single optimization step of the Levenberg-Marquardt method

- Compute $\mathbf{r}(\mathbf{x}_k)$, $\mathbf{J}(\mathbf{x}_k)$, $\mathbf{J}(\mathbf{x}_k)^{\mathsf{T}}\mathbf{J}(\mathbf{x}_k)$ and $-\nabla f(\mathbf{x}_k)$.
- do
  1. Solve
  $$\left[\mathbf{J}(\mathbf{x}_k)^{\mathsf{T}}\mathbf{J}(\mathbf{x}_k) + \lambda \mathbf{I}\right] \mathbf{d}_k = -\mathbf{J}(\mathbf{x}_k)^{\mathsf{T}}\mathbf{r}(\mathbf{x}_k).$$
  2. Compute $f(\mathbf{x}_k + \mathbf{d}_k)$.
  3. Compute
  $$\rho = \frac{f(\mathbf{x}_k) - f(\mathbf{x}_k + \mathbf{d}_k)}{f_k(\mathbf{0}) - f_k(\mathbf{d}_k)} = \frac{\text{actual decrease}}{\text{predicted decrease}}.$$
  4. If $\rho < \rho_{\min}$, then $\lambda = k\lambda$,
  else if $\rho > \rho_{\max}$, then $\lambda = \lambda/k$.

  while $\rho < 0$