Lecture 2
Algorithms for unconstrained nonlinear optimisation. Direct methods

Analysis and Development of Algorithms

УНИВЕРСИТЕТ ИТМО

Overview

Literature

- Bazara and Shetty. Nonlinear programming: Theory and algorithms
- Bazara, Jarvis and Sherali. Linear Programming and Network Flows
- Bonnans, Gilbert, Lemarechal, and Sagastizabal. Numerical Optimization. Theoretical and Practical Aspects
- Ben-Tal and Nemirovski. Lectures on Modern Convex Optimization: Analysis, Algorithms, and Engineering Applications
- Nocedal J., Wright Stephen J. Numerical Optimization
- and many others.

Optimisation problem

Optimisation methods are the methods for finding optimal (in some sense) solutions for mathematical models

Various applications, including in Machine Learning and Data Mining

A mathematical model can be usually expressed via an **objective function** $f = f(x)$, where x is generally a multidimensional vector, or via an optimality criterion (with constraints or without).

Optimisation problem

To solve the optimisation problem $f(x) \to \max_{x \in Q}(\min_{x \in Q})$ means to find $x^* \in Q$, where Q is the region of acceptability, such that f reaches the maximal (minimal) value at x^*. Notation: $x^* = \arg\max_{x \in Q}(\min_{x \in Q}) f(x)$.

If x^* is known, one can clearly find $f(x^*)$

- **Local** (simpler) and **global** (more complex) optimisation (coincide for some classes of f)
- The max-problem becomes the min-problem if one considers $-f$

Problem types. Unconstrained non-linear optimisation

Q, the region of acceptability, may be

- not defined (or reduced to not defined) (**unconstrained optimisation**); for example, minimise $f(x) = x^2$ for $x \in [-1, 1]$ (i.e. Q is defined) = minimise (without any constraints) the function

$$\tilde{f}(x) = \begin{cases} 1, & x \notin [-1, 1], \\ x^2, & x \in [-1, 1]. \end{cases}$$

- defined via a system S of linear or non-linear equations or inequalities (**constrained optimisation**)

$$f \text{ or } S \text{ (non-)linear } \rightarrow \begin{array}{l} \textbf{(non-)linear programming} \\ \textbf{(non-)linear optimisation} \end{array}$$

For purposes of Machine Learning and Data Mining, we are interested now in

unconstrained non-linear optimisation

Linear programming and constrained non-liner optimisation will be considered within the course projects

Methods for minimisation

Information about $f \rightarrow$ **direct**, **first-order** or **second-order** methods

Direct methods or zero-order methods

only values of f (but not those of its derivatives) are used.
☺ available for a wide class of functions; ☹ slow convergence

First-order methods

values of f and f' are used (gradient methods).
☺ relatively fast convergence; ☹ necessity to know the analytic expression of f and f'

Second-order methods

values of f, f' and f'' are used (Newtonian methods).
☺ fast convergence; ☹ necessity to know the analytic expression of f, f' and f''

One-dimensional direct methods

Exhaustive search (brute-force search)

Let $f(x) : [a, b] \to \mathbb{R}$. Approximately solve the optimisation problem $f(x) \to \min_{x \in [a, b]}$ by finding x^* with error $\varepsilon > 0$.

Algorithm

Consider the following point in $[a, b]$:

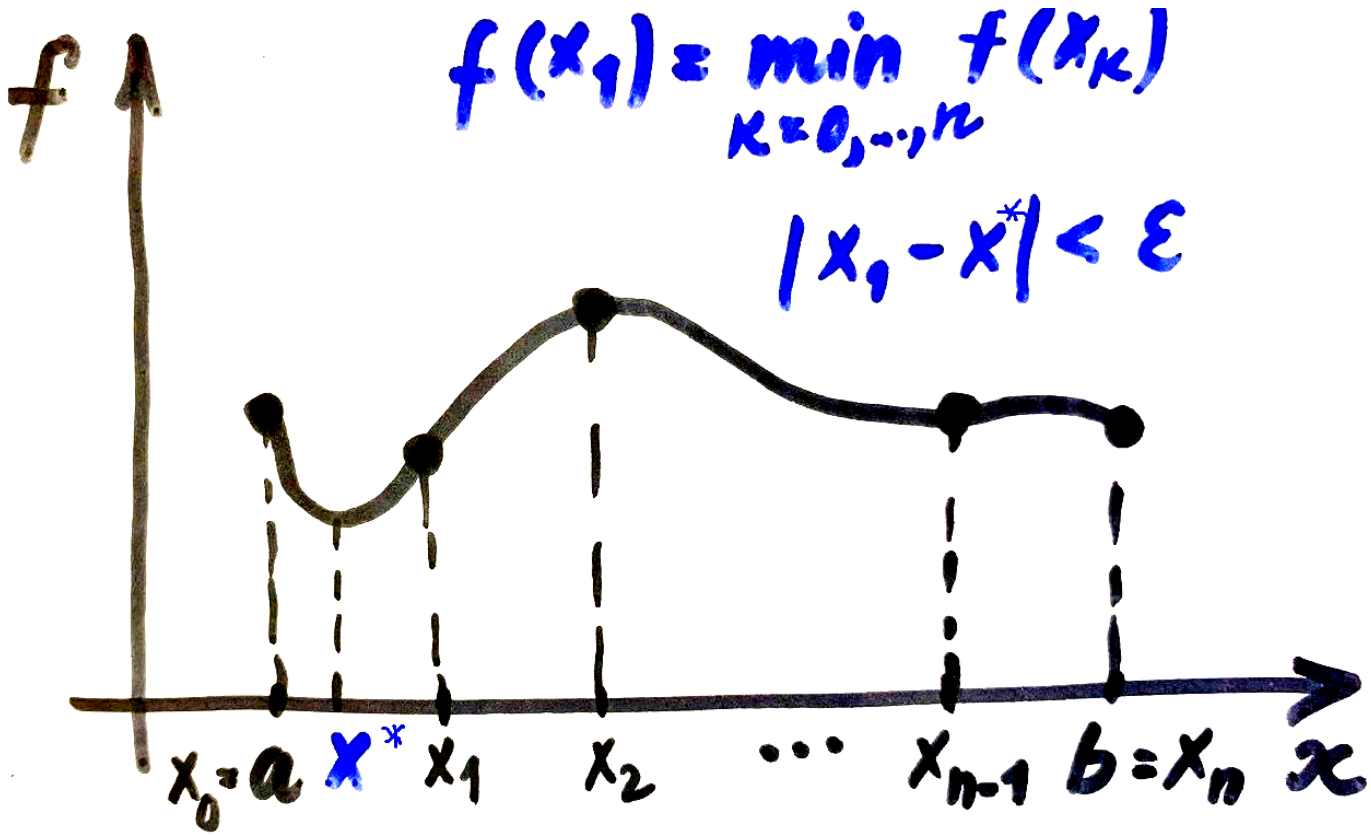
$$x_k = a + k\frac{b-a}{n}, \quad k = 0, \ldots, n,$$

where n is chosen so that $\frac{b-a}{n} \leqslant \varepsilon$.
Calculate $f(x_k)$ and find x_m, where $m \in \{1, \ldots, n\}$, such that

$$f(x_m) = \min_{k=0,\ldots,n} f(x_k).$$

Then $|x_m - x| < \varepsilon$. Use the obtained x_m as an approximant to x^*.

This method is very slow but may be sometimes used for finding initial approximations.


$f(x_1) = \min_{k=0,\dots,n} f(x_k)$
$|x_1 - x^*| < \varepsilon$
f
$x_0 = a$
x^*
x_1
x_2
$\dots$
x_{n-1}
$b = x_n$
x

Dichotomy method

Let $f(x) : [a_0, b_0] \to \mathbb{R}$ and be convex. Approximately solve the optimisation problem $f(x) \to \min_{x \in [a_0, b_0]}$ by finding x^* with error $\varepsilon > 0$.

Algorithm

Calculate

$$x_1 = \frac{a_0 + b_0 - \delta}{2}, \qquad x_2 = \frac{a_0 + b_0 + \delta}{2}, \qquad 0 < \delta < \varepsilon,$$

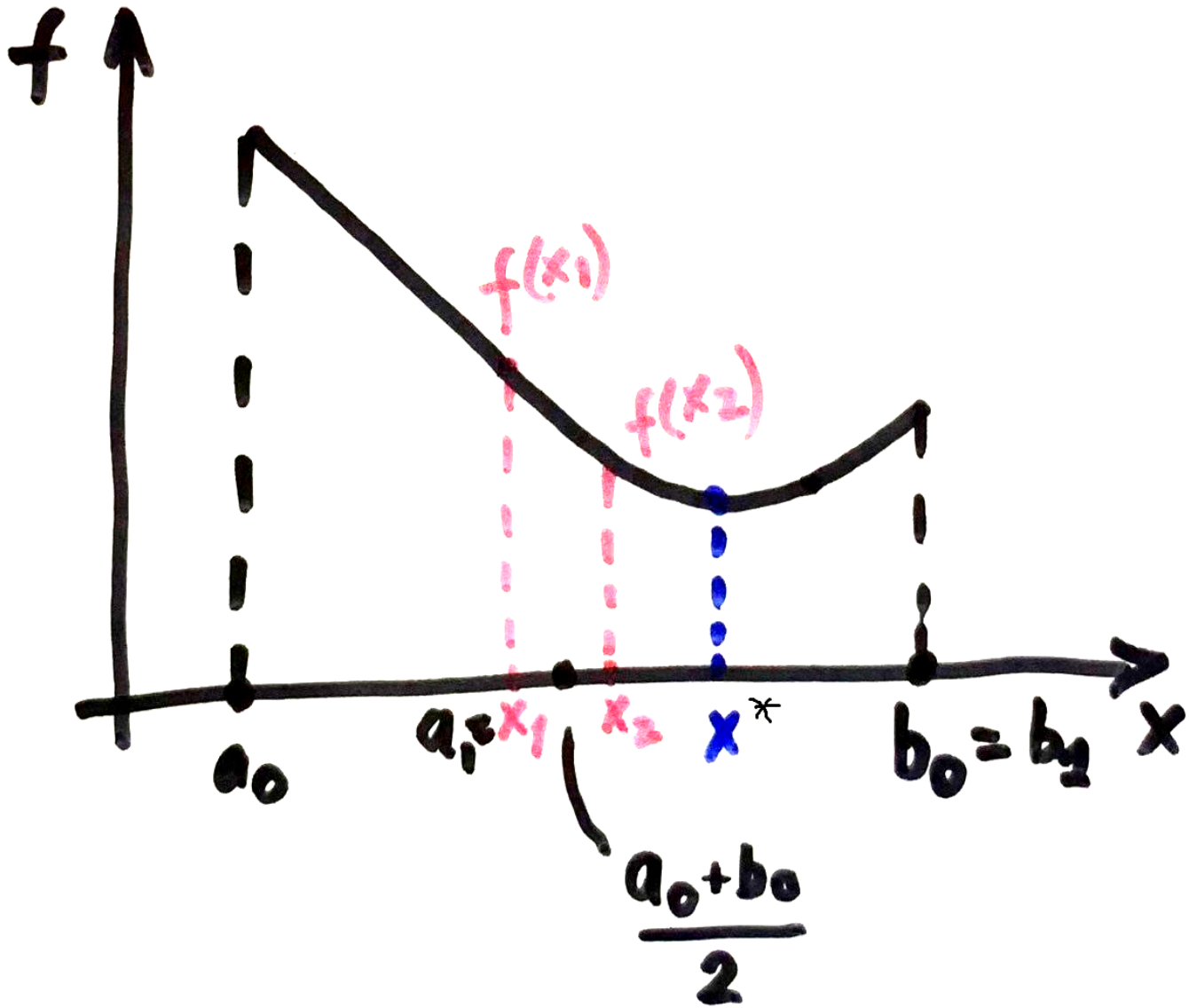
and $f(x_1)$ and $f(x_2)$.
Reduce the indeterminacy segment down to the segment $[a_1, b_1]$ as follows:

- if $f(x_1) \leqslant f(x_2)$, then $a_1 = a_0$ and $b_1 = x_2$;
- $a_1 = x_1$ and $b_1 = b_0$, otherwise.

Furthermore, by analogous formulas find x_1 and x_2 in the segment $[a_1, b_1]$ and repeat the reducing procedure down to a segment $[a_2, b_2]$, etc.
The search is stopped if at the current kth iteration it holds that

$$|a_k - b_k| < \varepsilon \qquad (x^* \in [a_k, b_k]).$$

If $\delta = 0$, then the dichotomy method is the bisection method.


f
f(x1)
f(x2)
a0
a1=x1
x2
x*
b0=b1
x
a0+b0
2

Golden section method

It allows to find the minimum in $[a_0, b_0]$ with less computations. Essentially, it is the dichotomy method with a special choice of δ.

Algorithm

Calculate

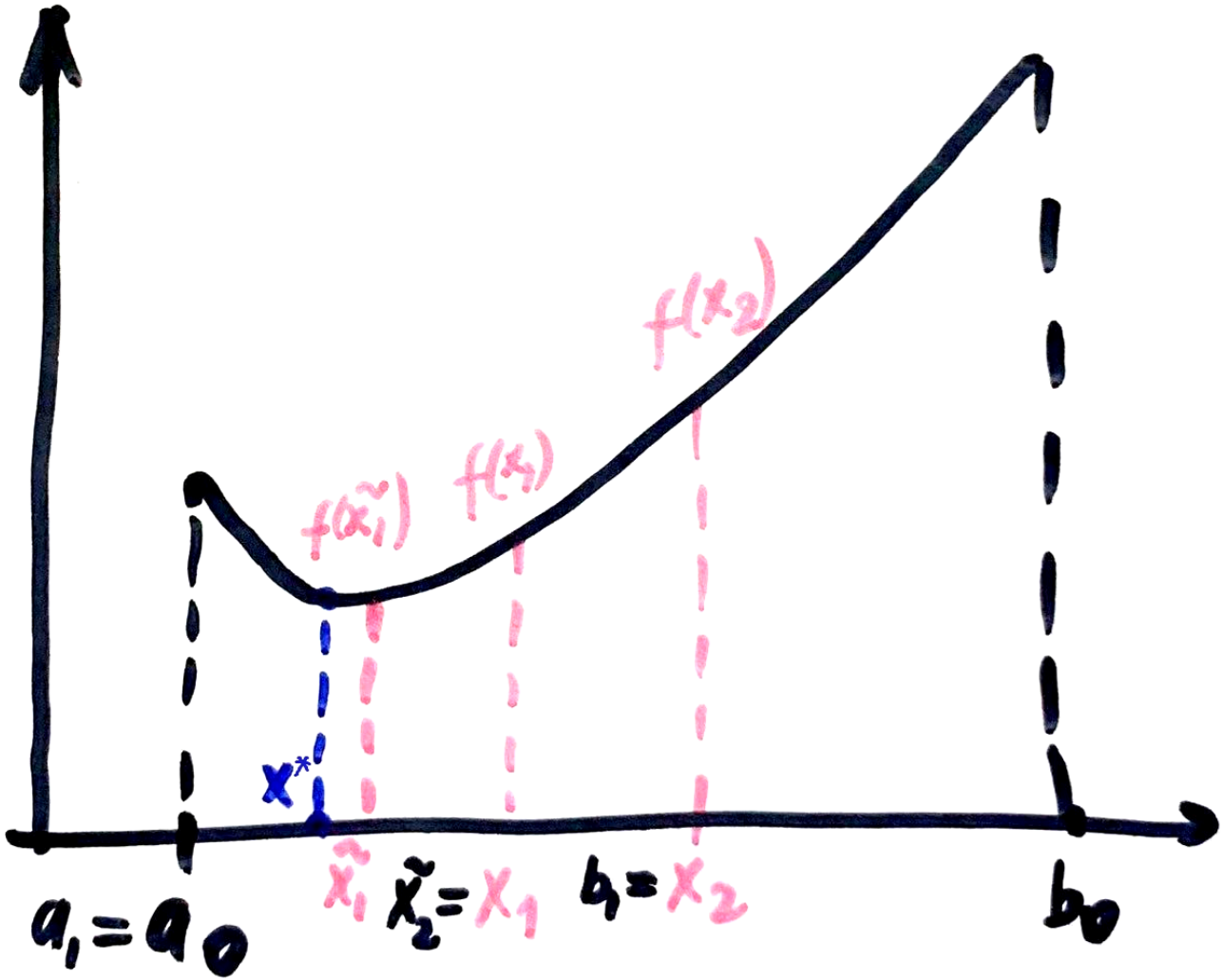
$$x_1 = a_0 + \frac{3-\sqrt{5}}{2}(b_0 - a_0), \quad x_2 = b_0 + \frac{\sqrt{5}-3}{2}(b_0 - a_0), \quad \left(\frac{x_1+x_2}{2} = \frac{a_0+b_0}{2}\right),$$

and $f(x_1)$ and $f(x_2)$ (the first iteration requires to calculate two points and two values of f).
Reduce the indeterminacy segment down to the segment $[a_1, b_1]$ as follows:

- if $f(x_1) \leqslant f(x_2)$, then $a_1 = a_0$, $b_1 = x_2$ and $x_2 = x_1$;
- $a_1 = x_1$, $b_1 = b_0$ and $x_1 = x_2$, otherwise.

On the forthcoming iterations, calculate one point and one corresponding value of f: in the former case x_1 and $f(x_1)$, and in the latter one x_2 and $f(x_2)$.
The search is stopped if at the current kth iteration it holds that

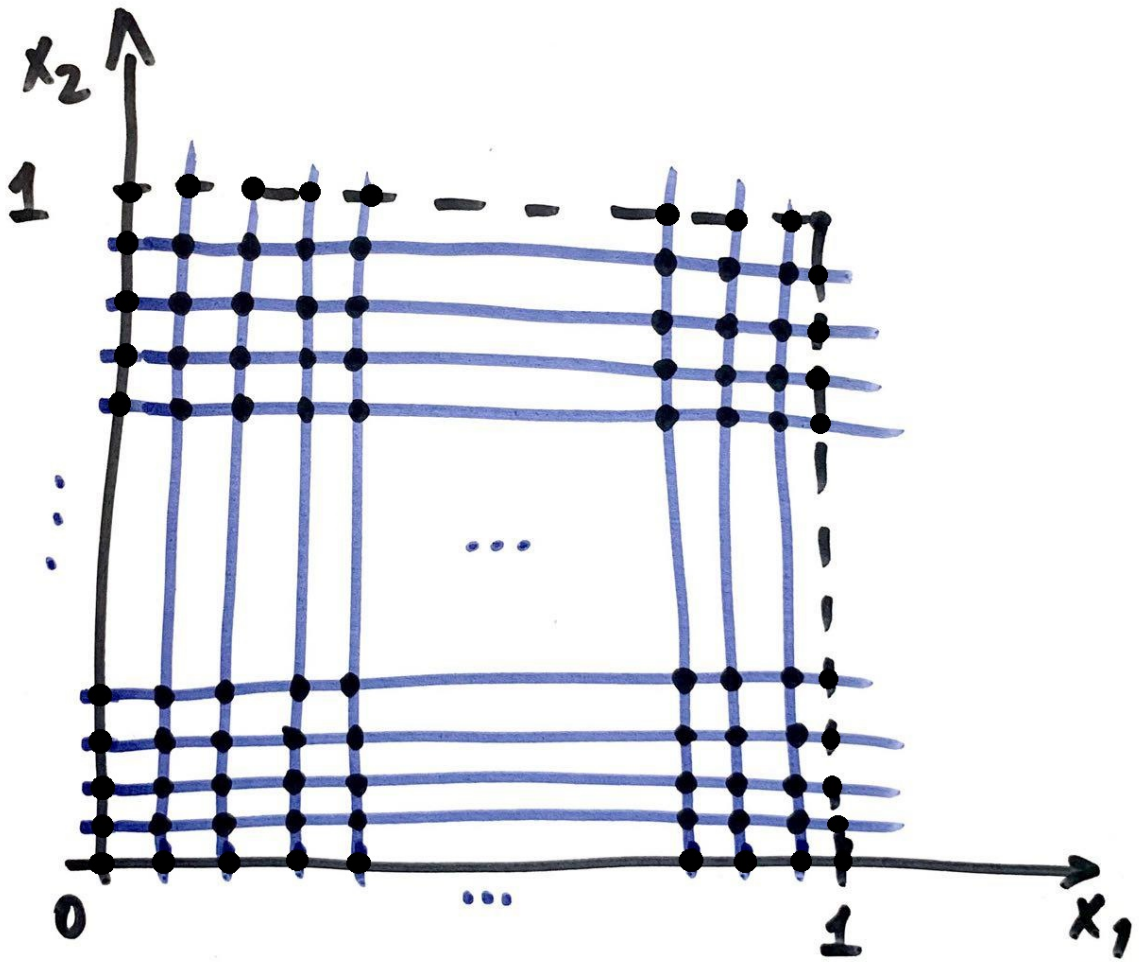
$$|a_k - b_k| < \varepsilon \qquad (x^* \in [a_k, b_k]).$$


f(x2)
f(x1~)
f(x1)
x*
a1 = a0
x1~
x2~ = x1
b1 = x2
b0

Multidimensional direct methods (the case of two variables)

Exhaustive search

Let $f(x): \ D \to \mathbb{R}$, where $x = (x_1, x_2)$, $D = \{[0,1] \times [0,1]\}$. Approximately solve the optimisation problem $f(x) \to \min_{(x,y) \in D}$ with error $\varepsilon > 0$.



Gauss method

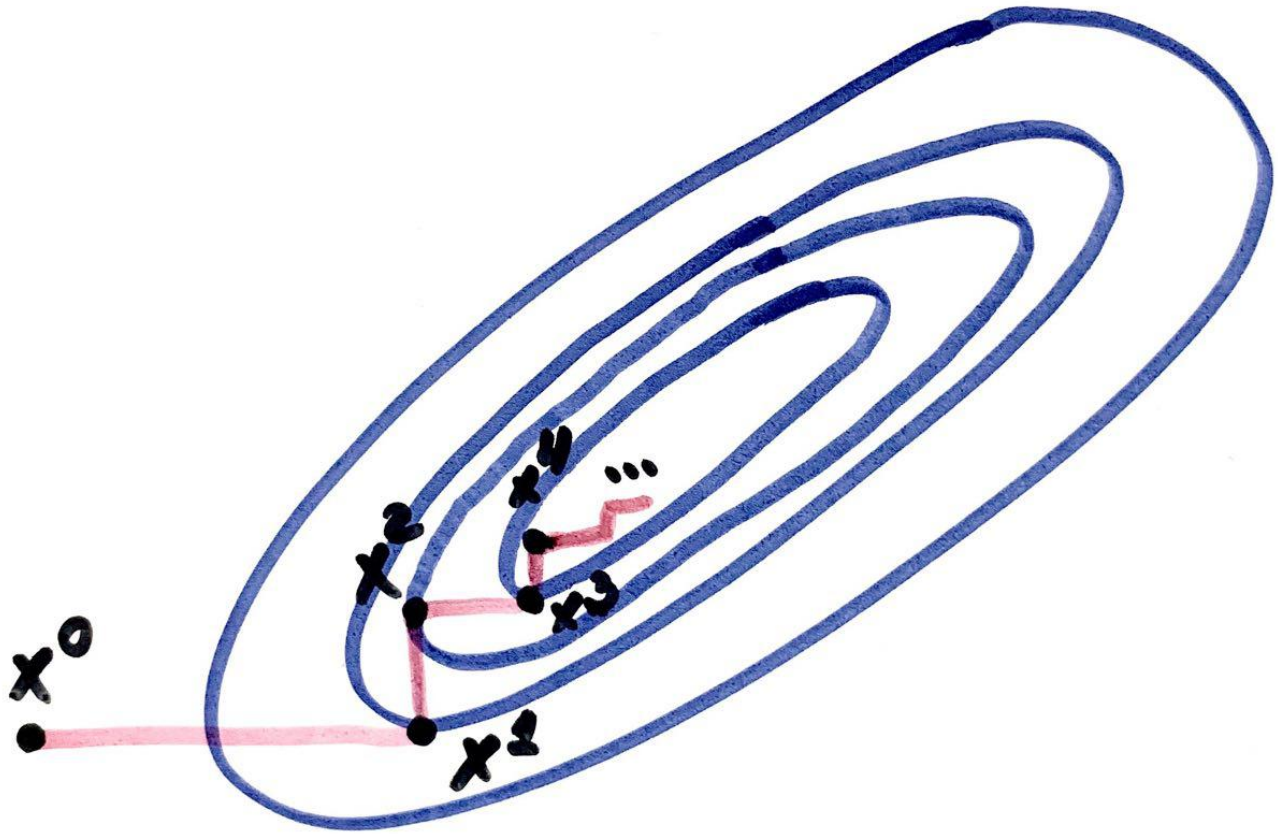
In each iteration, the minimisation is carried out only with respect to one vector component of the variable x. We consider $x = (x_1, x_2)$.

Algorithm

Let $x^0 = (x_1^0, x_2^0)$ be the initial approximation. In the first iteration, find the minimum point of f as a function of the first variable while others are fixed, i.e. $x_1^1 = \arg\min_{x_1} f(x_1, x_2^0)$, to get a new point $x^1 = (x_1^1, x_2^0)$.
Furthermore, using x^1, find the minimum point by varying only the second variable, i.e. find $x_2^1 = \arg\min_{x_2} f(x_1^1, x_2)$ and a new point $x^2 = (x_1^1, x_2^2)$.
Start searching again by the first variable, etc.
The search is stopped under one of the following criteria:

$$1) \quad |x_i^{k+1} - x_i^k| < \varepsilon, \quad i = 1, 2, \quad \text{or} \quad 2) \quad |f(x^{k+1}) - f(x^k)| < \varepsilon.$$

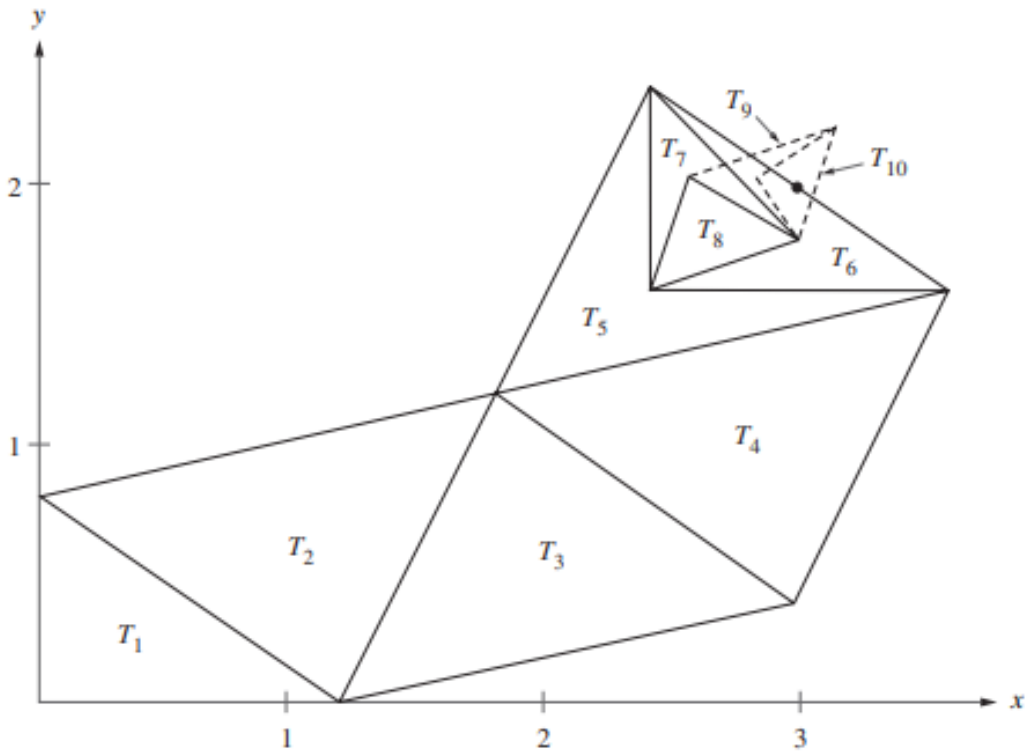
The method is simple but hardly efficient. Problems appear when the level lines of f are strongly elongated along the line $x_1 = x_2$. If the initial approximation is on $x_1 = x_2$, then the process gets stuck.


x0
x1
x2
x3
x4

Nelder-Mead method

For search, it uses simplexes in $\mathbb{R}^n$. In our case ($n = 2$), they are triangles.

This is an *heuristic approach* which can stuck in a local minima or converge to a non-stationary point.



Numerical Methods Using Matlab, 4th Edition, 2004
John H. Mathews and Kurtis K. Fink

Algorithm (of finding a minimal point of $f(x^{(1)}, x^{(2)})$)

Parameters: reflection coefficient $\alpha > 0$ (usually $\alpha = 1$), shrinking coefficient $\beta > 0$ (usually $\beta = 0.5$), dilatation coefficient $\gamma > 0$ (usually $\gamma = 2$).

Step 1 (preparation). Choose three points $x_i = (x_i^{(1)}, x_i^{(2)})$, $i = 1, 2, 3$ for the initial simplex (triangle in our case). Calculate $f_1 = f(x_1)$, $f_2 = f(x_2)$ and $f_3 = f(x_3)$.

Step 2 (sorting). Choose three points from the simplex vertices as follows: x_h with the largest value of f_h, x_g with the second-large value of f_g and x_l with the smallest value of f_l. The goal of the forthcoming manipulations is decreasing of f_h at least.

Step 3 (gravity centre). Find the gravity centre for all point except x_h: $x_c = \frac{1}{2} \sum_{i \neq h} x_i$.

Step 4 (reflection). Reflect the point x_h with respect to x_c with the coefficient α (for $\alpha = 1$ it is the central symmetry): $x_r = (1 + \alpha)x_c - \alpha x_h$. Calculate $f_r = f(x_r)$.

(to be continued)

Algorithm (continuation)

Step 5. Furthermore, check how the function values are decreased:

- If $f_r < f_l$, then the direction is right and we can dilate: calculate $x_e = (1-\gamma)x_c + \gamma x_r$ and $f_e = f(x_e)$.
- If $f_e < f_r$, the the simplex can be extended: set $x_h := x_e$ and go to Step 7.
- If $f_r < f_e$, then we moved too far: set $x_h := x_r$ and go to Step 7.
- If $f_l < f_r < f_g$, then the choice of the new point is good (the new one is better than previous two): set $x_h := x_r$ and go to Step 7.
- If $f_g < f_r < f_h$, then exchange x_r and x_h and f_r and f_h. After this, go to Step 6.
- If $f_h < f_r$, then go to Step 6.

As a result, $f_l < f_g < f_h < f_r$.

(to be continued)

Algorithm (continuation)

Step 6 (shrinking). Calculate $x_s = \beta x_h + (1-\beta)x_c$ and $f_s = f(x_s)$.

- If $f_s < f_h$, then set $x_h := x_s$ and go to Step 7.
- If $f_s > f_h$, then the initial points are the best.

Shrink the simplex globally as follows: $x_i := x_l + (x_i - x_l)/2$, $i \neq l$.

Step 7 (convergence check). Check the mutual closeness of the simplex vertices (for example, by estimating the deviation of the vertices), i.e. the closeness to the minimum point, as the method supposes.

If the required precision is not achieved, go to Step 2.

Example 1
Example 2

Thank you for your attention!