02610
Optimization and Data Fitting
Week 10: Derivative-Free Optimization

Yiqiu Dong

DTU Compute
Technical University of Denmark

9 November 2020

Optimization methods so far

$$\min_{\mathbf{x}\in\mathbb{R}^n} f(\mathbf{x}) \qquad f : \mathbb{R}^n \to \mathbb{R},\ f \in \mathcal{C}^2(\mathbb{R}^n)$$

The iteration step is essentially in the form of

$$\mathbf{x}_{k+1} = \mathbf{x}_k + \alpha_k \mathbf{p}_k,$$

and the search direction $\mathbf{p}_k$ typically is from solving a linear system

$$G_k \mathbf{p} = -\nabla f(\mathbf{x}_k).$$

Methods so far

$G_k = I,$	Steepest descent
$G_k = \nabla^2 f(\mathbf{x}_k),$	Newton
$G_k = B_k,$	Quasi-Newton

Why derivative-free optimization?

Some of the reasons to apply Derivative-Free Optimization (DFO):

- Growing sophistication of computer hardware and mathematical algorithms and software (which opens new possibilities for optimization).
- Function evaluations costly and noisy (one cannot trust derivatives or approximate them by finite differences).
- Binary codes (source code not available or owned by a company) ? making automatic differentiation impossible to apply.
- Legacy codes (written in the past and not maintained by the original authors).
- Lack of sophistication of the user (users need improvement but want to use something simple).

Finite-difference derivative approximations

- **Forward-difference:**

  Apply Taylor's theorem:

  $$f(\mathbf{x}+\mathbf{p}) = f(\mathbf{x}) + \nabla f(\mathbf{x})^T \mathbf{p} + O(\|\mathbf{p}\|_2^2), \quad \text{some } t \in (0,1).$$

  Set $\mathbf{p} = \epsilon \mathbf{e}_i$ with $\mathbf{e}_i$ as the ith unit vector, then $\nabla f(\mathbf{x})^T \mathbf{p} = \epsilon \frac{\partial f}{\partial x_i}$.

  Hence,

  $$\frac{\partial f}{\partial x_i}(\mathbf{x}) = \frac{f(\mathbf{x}+\epsilon \mathbf{e}_i) - f(\mathbf{x})}{\epsilon} + O(\epsilon).$$

Finite-difference derivative approximations

- **Central-difference:**

Apply Taylor's theorem:

$$f(\mathbf{x}+\mathbf{p}) = f(\mathbf{x}) + \nabla f(\mathbf{x})^T\mathbf{p} + \tfrac{1}{2}\mathbf{p}^T\nabla^2 f(\mathbf{x})\mathbf{p} + O(\|\mathbf{p}\|_2^3).$$

By setting $\mathbf{p} = \epsilon\mathbf{e}_i$ and $\mathbf{p} = -\epsilon\mathbf{e}_i$, respectively, we obtain,

$$f(\mathbf{x}+\epsilon\mathbf{e}_i) = f(\mathbf{x}) + \epsilon\frac{\partial f}{\partial x_i} + \tfrac{1}{2}\epsilon^2\frac{\partial^2 f}{\partial x_i^2} + O(\epsilon^3),$$
$$f(\mathbf{x}-\epsilon\mathbf{e}_i) = f(\mathbf{x}) - \epsilon\frac{\partial f}{\partial x_i} + \tfrac{1}{2}\epsilon^2\frac{\partial^2 f}{\partial x_i^2} + O(\epsilon^3).$$

Hence,

$$\frac{\partial f}{\partial x_i}(\mathbf{x}) = \frac{f(\mathbf{x}+\epsilon\mathbf{e}_k) - f(\mathbf{x}-\epsilon\mathbf{e}_k)}{2\epsilon} + O(\epsilon^2)$$

Noisy objective function

$$f(\mathbf{x}) = h(\mathbf{x}) + \phi(\mathbf{x}),$$

where h is a smooth function and ϕ represents the noise.

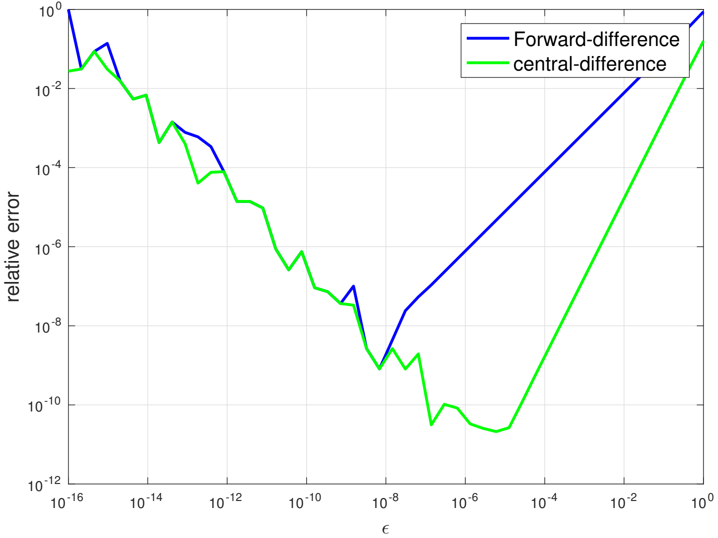
Apply the central-difference derivative approximation

$$\frac{\partial f}{\partial x_i}(\mathbf{x}) \approx \frac{\partial h}{\partial x_i}(\mathbf{x}) + \frac{\phi(\mathbf{x} + \epsilon \mathbf{e}_k) - \phi(\mathbf{x} - \epsilon \mathbf{e}_k)}{2\epsilon} + o(\epsilon^2)$$

If the noise dominates the difference interval ϵ, we cannot expect any accuracy at all in the finite-difference approximation.

Example: derivative approximations

Derivative of $f(x) = \sin x$ evaluated at $x = 1.0$



Automatic differentiation

- It converts the program into a sequence of simple elementary operations which have specified routines for computing derivatives.
- It applies chain rule to computer program.
- It is not symbolic differentiation.
- It is both efficient and numerical stable.
- Software: `http://www.autodiff.org`

Program: $f : \mathbb{R}^n \to \mathbb{R}$ expressed as m "simple" operations

$$
\begin{aligned}
v_1 &= h_1(x_1, x_2, \ldots, x_n) \\
v_2 &= h_2(x_1, x_2, \ldots, x_n, v_1) \\
v_3 &= h_3(x_1, x_2, \ldots, x_n, v_1, v_2) \\
&\vdots \\
v_m &= h_m(x_1, x_2, \ldots, x_n, v_1, v_2, \ldots, v_{m-1})
\end{aligned}
$$

such that $f(\mathbf{x}) = v_m$

Automatic differentiation – forward mode

Calculate the directional derivative in for $\mathbf{p}$

$$D_p v_i = \sum_{j=1}^{n} \frac{\partial v_i}{\partial x_j} p_j = \nabla_{\mathbf{x}} h_i(x_1, x_2, \ldots, x_n, v_1, v_2, \ldots, v_{i-1})^T \mathbf{p}$$

Example: Evaluate $\frac{\partial f}{\partial x_1}(\mathbf{x})$ where $f(\mathbf{x}) = x_1(1 + x_2 e^{x_1})^2$.

$$p = \begin{bmatrix} 1 \\ 0 \end{bmatrix} \quad \Rightarrow \quad D_p v_i = \frac{\partial v_i}{\partial x_1} = \dot{v}_i$$

function value:

$$\begin{aligned} v_1 &= \exp(x_1) \\ v_2 &= x_2 v_1 \\ v_3 &= v_2 + 1 \\ v_4 &= v_3^2 \\ v_5 &= x_1 v_4 \end{aligned}$$

directional derivative:

$$\begin{aligned} \dot{v}_1 &= v_1 \\ \dot{v}_2 &= x_2 \dot{v}_1 \\ \dot{v}_3 &= \dot{v}_2 \\ \dot{v}_4 &= 2 v_3 \dot{v}_3 \\ \dot{v}_5 &= \dot{v}_4 x_1 + v_4 \end{aligned}$$

Automatic differentiation – reverse mode

Define the adjoint variables $\bar{v}_i = \partial v_m / \partial v_i$ and $\frac{\partial f}{\partial x_i}(\mathbf{x}) = \partial v_m / \partial x_i$

$$\bar{v}_i = \sum_{j=i+1}^{m} \bar{v}_j \frac{\partial v_j}{\partial v_i}, \qquad \frac{\partial f}{\partial x_i}(\mathbf{x}) = \sum_{j=1}^{m} \bar{v}_j \frac{\partial v_j}{\partial x_i}$$

Example: Evaluate $\nabla f(\mathbf{x})$ where $f(\mathbf{x}) = x_1(1 + x_2 e^{x_1})^2$

function value:

$$\begin{aligned} v_1 &= \exp(x_1) \\ v_2 &= x_2 v_1 \\ v_3 &= v_2 + 1 \\ v_4 &= v_3^2 \\ v_5 &= x_1 v_4 \end{aligned}$$

Automatic differentiation – reverse mode (cont.)

Recursive definition of adjoint variables

$$
\begin{aligned}
\bar{v}_5 &= 1 \\
\bar{v}_4 &= \bar{v}_5 \frac{\partial v_5}{\partial v_4} \\
\bar{v}_3 &= \bar{v}_4 \frac{\partial v_4}{\partial v_3} + \bar{v}_5 \frac{\partial v_5}{\partial v_3} \\
\bar{v}_2 &= \bar{v}_3 \frac{\partial v_3}{\partial v_2} + \bar{v}_4 \frac{\partial v_4}{\partial v_2} + \bar{v}_5 \frac{\partial v_5}{\partial v_2} \\
\bar{v}_1 &= \bar{v}_2 \frac{\partial v_2}{\partial v_1} + \bar{v}_3 \frac{\partial v_3}{\partial v_1} + \bar{v}_4 \frac{\partial v_4}{\partial v_1} + \bar{v}_5 \frac{\partial v_5}{\partial v_1} \\
\frac{\partial f}{\partial x_1}(\mathbf{x}) &= \bar{v}_1 \frac{\partial v_1}{\partial x_1} + \bar{v}_2 \frac{\partial v_2}{\partial x_1} + \bar{v}_3 \frac{\partial v_3}{\partial x_1} + \bar{v}_4 \frac{\partial v_4}{\partial x_1} + \bar{v}_5 \frac{\partial v_5}{\partial x_1} \\
\frac{\partial f}{\partial x_2}(\mathbf{x}) &= \bar{v}_1 \frac{\partial v_1}{\partial x_2} + \bar{v}_2 \frac{\partial v_2}{\partial x_2} + \bar{v}_3 \frac{\partial v_3}{\partial x_2} + \bar{v}_4 \frac{\partial v_4}{\partial x_2} + \bar{v}_5 \frac{\partial v_5}{\partial x_2}
\end{aligned}
$$

Automatic differentiation – reverse mode (cont.)

Start with $v_5 = x_1 v_4$

$$
\begin{aligned}
\bar{v}_5 &= 1 \\
\bar{v}_4 &= \bar{v}_5 \frac{\partial v_5}{\partial v_4} \\
\bar{v}_3 &= \bar{v}_4 \frac{\partial v_4}{\partial v_3} + \bar{v}_5 \frac{\partial v_5}{\partial v_3} \\
\bar{v}_2 &= \bar{v}_3 \frac{\partial v_3}{\partial v_2} + \bar{v}_4 \frac{\partial v_4}{\partial v_2} + \bar{v}_5 \frac{\partial v_5}{\partial v_2} \\
\bar{v}_1 &= \bar{v}_2 \frac{\partial v_2}{\partial v_1} + \bar{v}_3 \frac{\partial v_3}{\partial v_1} + \bar{v}_4 \frac{\partial v_4}{\partial v_1} + \bar{v}_5 \frac{\partial v_5}{\partial v_1} \\
\frac{\partial f}{\partial x_1}(\mathbf{x}) &= \bar{v}_1 \frac{\partial v_1}{\partial x_1} + \bar{v}_2 \frac{\partial v_2}{\partial x_1} + \bar{v}_3 \frac{\partial v_3}{\partial x_1} + \bar{v}_4 \frac{\partial v_4}{\partial x_1} + \bar{v}_5 \frac{\partial v_5}{\partial x_1} \\
\frac{\partial f}{\partial x_2}(\mathbf{x}) &= \bar{v}_1 \frac{\partial v_1}{\partial x_2} + \bar{v}_2 \frac{\partial v_2}{\partial x_2} + \bar{v}_3 \frac{\partial v_3}{\partial x_2} + \bar{v}_4 \frac{\partial v_4}{\partial x_2} + \bar{v}_5 \frac{\partial v_5}{\partial x_2}
\end{aligned}
$$

Automatic differentiation – reverse mode (cont.)

Start with $v_5 = x_1 v_4$

$$
\begin{aligned}
\bar{v}_5 &= 1 \\
\bar{v}_4 &= {\color{red}x_1} \\
\bar{v}_3 &= \bar{v}_4 \frac{\partial v_4}{\partial v_3} + {\color{red}0} \\
\bar{v}_2 &= \bar{v}_3 \frac{\partial v_3}{\partial v_2} + \bar{v}_4 \frac{\partial v_4}{\partial v_2} + {\color{red}0} \\
\bar{v}_1 &= \bar{v}_2 \frac{\partial v_2}{\partial v_1} + \bar{v}_3 \frac{\partial v_3}{\partial v_1} + \bar{v}_4 \frac{\partial v_4}{\partial v_1} + {\color{red}0} \\
\frac{\partial f}{\partial x_1}(\mathbf{x}) &= \bar{v}_1 \frac{\partial v_1}{\partial x_1} + \bar{v}_2 \frac{\partial v_2}{\partial x_1} + \bar{v}_3 \frac{\partial v_3}{\partial x_1} + \bar{v}_4 \frac{\partial v_4}{\partial x_1} + {\color{red}v_4} \\
\frac{\partial f}{\partial x_2}(\mathbf{x}) &= \bar{v}_1 \frac{\partial v_1}{\partial x_2} + \bar{v}_2 \frac{\partial v_2}{\partial x_2} + \bar{v}_3 \frac{\partial v_3}{\partial x_2} + \bar{v}_4 \frac{\partial v_4}{\partial x_2} + {\color{red}0}
\end{aligned}
$$

Automatic differentiation – reverse mode (cont.)

Continue with $v_4 = v_3^2$

$$
\begin{aligned}
\bar{v}_5 &= 1 \\
\bar{v}_4 &= x_1 \\
\bar{v}_3 &= \bar{v}_4 \frac{\partial v_4}{\partial v_3} \\
\bar{v}_2 &= \bar{v}_3 \frac{\partial v_3}{\partial v_2} + \bar{v}_4 \frac{\partial v_4}{\partial v_2} \\
\bar{v}_1 &= \bar{v}_2 \frac{\partial v_2}{\partial v_1} + \bar{v}_3 \frac{\partial v_3}{\partial v_1} + \bar{v}_4 \frac{\partial v_4}{\partial v_1} \\
\frac{\partial f}{\partial x_1}(\mathbf{x}) &= \bar{v}_1 \frac{\partial v_1}{\partial x_1} + \bar{v}_2 \frac{\partial v_2}{\partial x_1} + \bar{v}_3 \frac{\partial v_3}{\partial x_1} + \bar{v}_4 \frac{\partial v_4}{\partial x_1} + v_4 \\
\frac{\partial f}{\partial x_2}(\mathbf{x}) &= \bar{v}_1 \frac{\partial v_1}{\partial x_2} + \bar{v}_2 \frac{\partial v_2}{\partial x_2} + \bar{v}_3 \frac{\partial v_3}{\partial x_2} + \bar{v}_4 \frac{\partial v_4}{\partial x_2}
\end{aligned}
$$

Automatic differentiation – reverse mode (cont.)

Continue with $v_4 = v_3^2$

$$
\begin{aligned}
\bar{v}_5 &= 1 \\
\bar{v}_4 &= x_1 \\
\bar{v}_3 &= 2\bar{v}_4 v_3 \\
\bar{v}_2 &= \bar{v}_3 \frac{\partial v_3}{\partial v_2} + 0 \\
\bar{v}_1 &= \bar{v}_2 \frac{\partial v_2}{\partial v_1} + \bar{v}_3 \frac{\partial v_3}{\partial v_1} + 0 \\
\frac{\partial f}{\partial x_1}(\mathbf{x}) &= \bar{v}_1 \frac{\partial v_1}{\partial x_1} + \bar{v}_2 \frac{\partial v_2}{\partial x_1} + \bar{v}_3 \frac{\partial v_3}{\partial x_1} + 0 + v_4 \\
\frac{\partial f}{\partial x_2}(\mathbf{x}) &= \bar{v}_1 \frac{\partial v_1}{\partial x_2} + \bar{v}_2 \frac{\partial v_2}{\partial x_2} + \bar{v}_3 \frac{\partial v_3}{\partial x_2} + 0
\end{aligned}
$$

Automatic differentiation – reverse mode (cont.)

Continue with $v_3 = v_2 + 1$

$$
\begin{aligned}
\bar{v}_5 &= 1 \\
\bar{v}_4 &= x_1 \\
\bar{v}_3 &= 2\bar{v}_4 v_3 \\
\bar{v}_2 &= \bar{v}_3 \frac{\partial v_3}{\partial v_2} \\
\bar{v}_1 &= \bar{v}_2 \frac{\partial v_2}{\partial v_1} + \bar{v}_3 \frac{\partial v_3}{\partial v_1} \\
\frac{\partial f}{\partial x_1}(\mathbf{x}) &= \bar{v}_1 \frac{\partial v_1}{\partial x_1} + \bar{v}_2 \frac{\partial v_2}{\partial x_1} + \bar{v}_3 \frac{\partial v_3}{\partial x_1} + v_4 \\
\frac{\partial f}{\partial x_2}(\mathbf{x}) &= \bar{v}_1 \frac{\partial v_1}{\partial x_2} + \bar{v}_2 \frac{\partial v_2}{\partial x_2} + \bar{v}_3 \frac{\partial v_3}{\partial x_2}
\end{aligned}
$$

Automatic differentiation – reverse mode (cont.)

Continue with $v_3 = v_2 + 1$

$$
\begin{aligned}
\bar{v}_5 &= 1 \\
\bar{v}_4 &= x_1 \\
\bar{v}_3 &= 2\bar{v}_4 v_3 \\
\bar{v}_2 &= {\color{red}\bar{v}_3} \\
\bar{v}_1 &= \bar{v}_2 \frac{\partial v_2}{\partial v_1} + {\color{red}0} \\
\frac{\partial f}{\partial x_1}(\mathbf{x}) &= \bar{v}_1 \frac{\partial v_1}{\partial x_1} + \bar{v}_2 \frac{\partial v_2}{\partial x_1} + {\color{red}0} + v_4 \\
\frac{\partial f}{\partial x_2}(\mathbf{x}) &= \bar{v}_1 \frac{\partial v_1}{\partial x_2} + \bar{v}_2 \frac{\partial v_2}{\partial x_2} + {\color{red}0}
\end{aligned}
$$

Automatic differentiation – reverse mode (cont.)

Continue with $v_2 = x_2 v_1$

$$
\begin{aligned}
\bar{v}_5 &= 1 \\
\bar{v}_4 &= x_1 \\
\bar{v}_3 &= 2\bar{v}_4 v_3 \\
\bar{v}_2 &= \bar{v}_3 \\
\bar{v}_1 &= \bar{v}_2 \frac{\partial v_2}{\partial v_1} \\
\frac{\partial f}{\partial x_1}(\mathbf{x}) &= \bar{v}_1 \frac{\partial v_1}{\partial x_1} + \bar{v}_2 \frac{\partial v_2}{\partial x_1} + v_4 \\
\frac{\partial f}{\partial x_2}(\mathbf{x}) &= \bar{v}_1 \frac{\partial v_1}{\partial x_2} + \bar{v}_2 \frac{\partial v_2}{\partial x_2}
\end{aligned}
$$

Automatic differentiation – reverse mode (cont.)

Continue with $v_2 = x_2 v_1$

$$
\begin{aligned}
\bar{v}_5 &= 1 \\
\bar{v}_4 &= x_1 \\
\bar{v}_3 &= 2\bar{v}_4 v_3 \\
\bar{v}_2 &= \bar{v}_3 \\
\bar{v}_1 &= \bar{v}_2 x_2 \\
\frac{\partial f}{\partial x_1}(\mathbf{x}) &= \bar{v}_1 \frac{\partial v_1}{\partial x_1} + 0 + v_4 \\
\frac{\partial f}{\partial x_2}(\mathbf{x}) &= \bar{v}_1 \frac{\partial v_1}{\partial x_2} + \bar{v}_2 v_1
\end{aligned}
$$

Automatic differentiation – reverse mode (cont.)

And finally $v_1 = \exp(x_1)$

$$
\begin{aligned}
\bar{v}_5 &= 1 \\
\bar{v}_4 &= x_1 \\
\bar{v}_3 &= 2\bar{v}_4 v_3 \\
\bar{v}_2 &= \bar{v}_3 \\
\bar{v}_1 &= \bar{v}_2 x_2 \\
\frac{\partial f}{\partial x_1}(\mathbf{x}) &= \bar{v}_1 \frac{\partial v_1}{\partial x_1} + v_4 \\
\frac{\partial f}{\partial x_2}(\mathbf{x}) &= \bar{v}_1 \frac{\partial v_1}{\partial x_2} + \bar{v}_2 v_1
\end{aligned}
$$

Automatic differentiation – reverse mode (cont.)

And finally $v_1 = \exp(x_1)$

$$\begin{aligned}
\bar{v}_5 &= 1 \\
\bar{v}_4 &= x_1 \\
\bar{v}_3 &= 2\bar{v}_4 v_3 \\
\bar{v}_2 &= \bar{v}_3 \\
\bar{v}_1 &= \bar{v}_2 x_2 \\
\frac{\partial f}{\partial x_1}(\mathbf{x}) &= \bar{v}_1 v_1 + v_4 \\
\frac{\partial f}{\partial x_2}(\mathbf{x}) &= 0 + \bar{v}_2 v_1
\end{aligned}$$

Automatic differentiation – reverse mode (cont.)

gradient is $\nabla f(x) = \begin{bmatrix} \bar{x}_1 \\ \bar{x}_2 \end{bmatrix}$

$$\begin{aligned}
\bar{v}_5 &= 1 \\
\bar{v}_4 &= x_1 \\
\bar{v}_3 &= 2\bar{v}_4 v_3 \\
\bar{v}_2 &= \bar{v}_3 \\
\bar{v}_1 &= \bar{v}_2 x_2 \\
\frac{\partial f}{\partial x_1}(\mathbf{x}) &= \bar{v}_1 v_1 + v_4 \\
\frac{\partial f}{\partial x_2}(\mathbf{x}) &= \bar{v}_2 v_1
\end{aligned}$$

Derivative-free methods

- **(Directional) Direct search methods**
  - Achieve descent by using positive bases or positive spanning sets and moving in the directions of the best points (in patterns or meshes).
  - Only use function values, and do not estimate derivative.
  - **Examples:** coordinate search method, pattern-search methods, etc.
- **(Simplicial) Direct search methods**
  - Ensure descent from simplex operations like reflections, by moving in the direction away from the point with the worst function value.
  - Only use function values, and do not estimate derivative.
  - **Examples:** Nelder-Mead method and its modifications.
- **Line-search methods**
  - Compute gradient approximation
  - Perform inaccurate line-search
  - **Examples:** implicit filtering method.
- **Trust-region methods**
  - Minimize trust-region subproblems defined by fully-linear or fully-quadratic models (typically built from interpolation or regression).
  - **Examples:** model-based methods.

Coordinate search method

Consider the unconstrained problem

$$\min_{\mathbf{x}\in\mathbb{R}^n} f(\mathbf{x}) \qquad f:\mathbb{R}^n \to \mathbb{R}.$$

- **Initialization:**
  $\mathbf{x}_0$: starting point in $\mathbb{R}^n$ such that $f(\mathbf{x}_0) < \infty$.
  γ_0: initial step length.
- **Iteration step:** for $k = 0, 1, \cdots$
  If $f(\mathbf{t}) < f(\mathbf{x}_k)$ for some $\mathbf{t} \in \mathcal{D}(\mathbf{x}_k, \gamma_k) = \{\mathbf{x}_k \pm \gamma_k \mathbf{e}_i : i = 1, \cdots, n\}$,
  set $\mathbf{x}_{k+1} = \mathbf{t}$,
  and $\gamma_{k+1} = \gamma_k$;
  otherwise $\mathbf{x}_k$ is a local minimum with respect to $\mathcal{D}(\mathbf{x}_k, \gamma_k)$
  Set $\mathbf{x}_{k+1} = \mathbf{x}_k$,
  and $\gamma_{k+1} = \frac{1}{2}\gamma_k$.

Coordinate search method

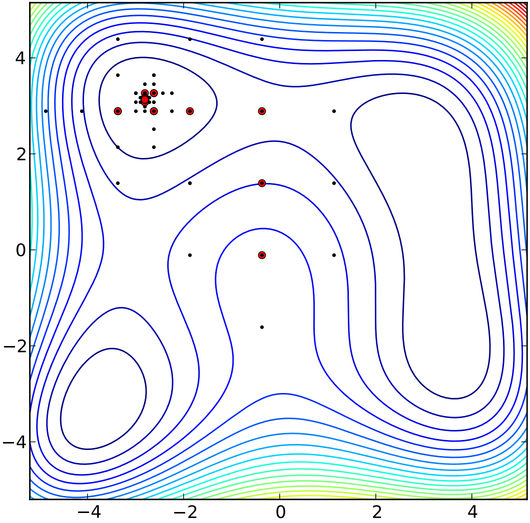
Consider the unconstrained problem

$$\min_{\mathbf{x}\in\mathbb{R}^n} f(\mathbf{x}) \qquad f : \mathbb{R}^n \to \mathbb{R}.$$

- **Initialization:**
  $\mathbf{x}_0$: starting point in $\mathbb{R}^n$ such that $f(\mathbf{x}_0) < \infty$.
  γ_0: initial step length.
- **Iteration step:** for $k = 0, 1, \cdots$
  If $f(\mathbf{t}) < f(\mathbf{x}_k)$ for some $\mathbf{t} \in \mathcal{D}(\mathbf{x}_k, \gamma_k) = \{\mathbf{x}_k \pm \gamma_k \mathbf{e}_i : i = 1, \cdots, n\}$,
  set $\mathbf{x}_{k+1} = \mathbf{t}$,
  and $\gamma_{k+1} = \gamma_k$;
  otherwise $\mathbf{x}_k$ is a local minimum with respect to $\mathcal{D}(\mathbf{x}_k, \gamma_k)$
  Set $\mathbf{x}_{k+1} = \mathbf{x}_k$,
  and $\gamma_{k+1} = \frac{1}{2}\gamma_k$.

Example: Coordinate search

Himmelblau's function: $f(x_1, x_2) = (x_1^2 + x_2 - 11)^2 + (x_1 + x_2^2 - 7)^2$



Pattern-search methods

The pattern-search method generalizes coordinate search in that it allows the use of a richer set of search directions at each direction. These search directions with a given step length form a "frame" or "stencil" around the current iterate.

- **Initialization:**
  $\mathbf{x}_0$: starting point in $\mathbb{R}^n$ such that $f(\mathbf{x}_0) < \infty$.
  γ_0: initial step length.
  $\mathcal{D}_0$: initial direction set.
  $\rho : [0, \infty) \to \mathbb{R}$: sufficient decrease function (a increasing function)
- **Iteration step:** for $k = 0, 1, \cdots$
  If $f(\mathbf{x}_k + \gamma_k \mathbf{p}_i) < f(\mathbf{x}_k) - \rho(\gamma_k)$ for some $\mathbf{p}_k \in \mathcal{D}_k$,
  set $\mathbf{x}_{k+1} = \mathbf{x}_k + \gamma_k \mathbf{p}_k$ for such $\mathbf{p}_k$,
  and $\gamma_{k+1} = \phi_k \gamma_k$ for some $\phi_k \geq 1$;
  else
  Set $\mathbf{x}_{k+1} = \mathbf{x}_k$,
  and $\gamma_{k+1} = \theta_k \gamma_k$ for some $0 < \theta_k < 1$.

Pattern-search methods

- Two conditions for $\mathcal{D}_k$:
  1. $\kappa(\mathcal{D}_k) = \min_{\mathbf{v}\in\mathbb{R}^n} \max_{\mathbf{p}\in\mathcal{D}_k} \frac{\mathbf{v}^T\mathbf{p}}{\|\mathbf{v}\|_2\|\mathbf{p}\|_2} \geq \delta$. It is inspired by $\cos(\theta) = \frac{-\nabla f_k^T \mathbf{p}}{\|\nabla f_k\|_2\|\mathbf{p}\|_2}$.
  2. The lengths of all vectors in $\mathcal{D}_k$ are roughly similar, so that the diameter of the frame is captured by γ_k.
- The coordinate search method is a special case of the pattern-search methods with $\mathcal{D}_k = \{\mathbf{e}_1, -\mathbf{e}_1, \cdots, \mathbf{e}_n, -\mathbf{e}_n\}$. Further, $\kappa(\mathcal{D}_k) = 0$ for all k.
- The method may exit search as soon as a better point is found in order to save on function evaluations.
- It converges to stationary point if $f(\mathbf{x})$ is continuously differentiable.

Simplex

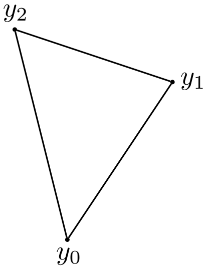
Nondegenerate *n*-simplex

Consider $n+1$ points $\mathbf{y}_0, \mathbf{y}_1, \ldots, \mathbf{y}_n \in \mathbb{R}^n$ such that

$$\mathbf{y}_1 - \mathbf{y}_0, \mathbf{y}_2 - \mathbf{y}_0, \ldots, \mathbf{y}_n - \mathbf{y}_0$$

are linearly independent. Then, its convex hull forms a nondegenerate n-simplex.

Example: Two-dimensional simplex in $\mathbb{R}^2$



Simplex

Let Y be a nondegenerate n-simplex with vertices $\mathbf{y}_0, \mathbf{y}_1, \ldots, \mathbf{y}_n$ in $\mathbb{R}^n$ and

$$M(Y) = [\mathbf{y}_1 - \mathbf{y}_0 \quad \mathbf{y}_2 - \mathbf{y}_0 \quad \cdots \quad \mathbf{y}_n - \mathbf{y}_0]$$

diameter:

$$\mathbf{diam}(Y) = \max_{0 \le i < j \le n} \|\mathbf{y}_i - \mathbf{y}_j\|$$

volume:

$$\mathbf{vol}(Y) = \frac{|\det(M(Y))|}{n!},$$

normalized volume:

$$\mathbf{von}(Y) = \frac{|\det(M(Y))|}{n!\,\mathbf{diam}(Y)^n}$$

Nelder-Mead method

- It was proposed by Nelder & Mead in 1965.
- It is also known as "downhill simplex method" and "amoeba method".

Outline: Start with nondegenerate n-simplex in $\mathbb{R}^n$; at each iteration,

1. order $\{\mathbf{y}_0, \mathbf{y}_1, \ldots, \mathbf{y}_n\}$ such that

$$f_0 \leq f_1 \leq \ldots \leq f_n, \quad f_i = f(\mathbf{y}_i).$$

2. compute centroid of the best n points

$$\mathbf{y}_\mathrm{c} = \frac{1}{n} \sum_{i=0}^{n-1} \mathbf{y}_i$$

3. either shrink simplex or replace worst point $\mathbf{y}_n$ by

$$\mathbf{y}(\delta) = \mathbf{y}_\mathrm{c} + \delta(\mathbf{y}_\mathrm{c} - \mathbf{y}_n)$$

Nelder-Mead method

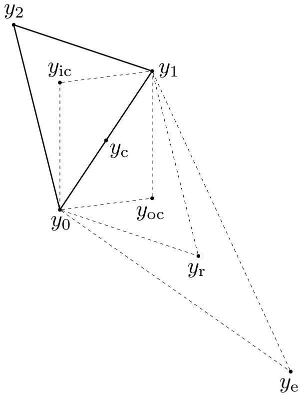
Step 3: Replace $\mathbf{y}_n$

- reflection: $\mathbf{y}_{\mathrm{r}} = \mathbf{y}(1)$
- expansion: $\mathbf{y}_{\mathrm{e}} = \mathbf{y}(2)$
- inner contraction: $\mathbf{y}_{\mathrm{ic}} = \mathbf{y}(-0.5)$
- outer contraction: $\mathbf{y}_{\mathrm{oc}} = \mathbf{y}(0.5)$

or shrink simplex: replace $\mathbf{y}_1, \ldots, \mathbf{y}_n$ by

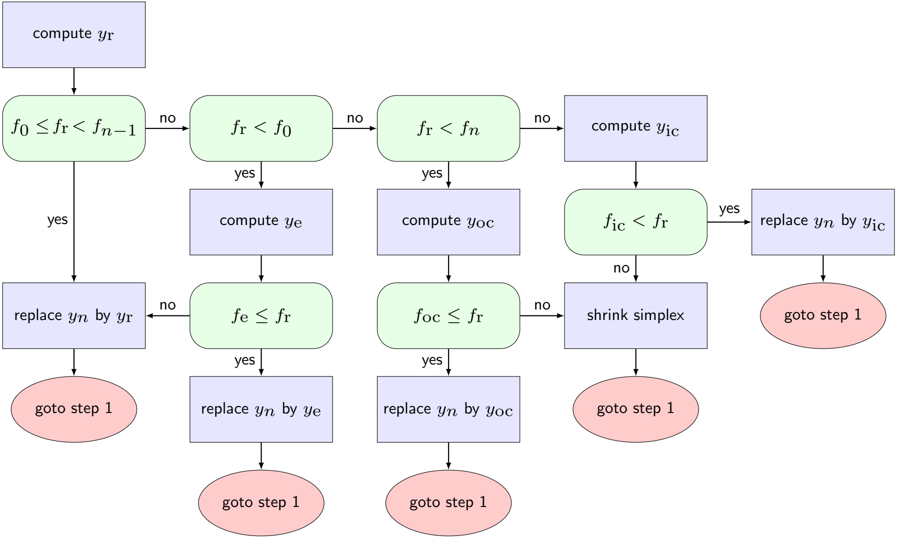
$$\lambda \mathbf{y}_i + (1-\lambda)\mathbf{y}_0, \quad i = 1, \ldots, n$$

(typically, $\lambda = 1/2$)



Nelder-Mead method

Given: $\mathbf{y}_0, \mathbf{y}_1, \ldots, \mathbf{y}_n$ and function values $f_0 \le f_1 \le \ldots \le f_n$



Nelder-Mead method

Let $f : \mathbb{R}^n \to \mathbb{R}$ be bounded from below

- Shrink steps are never performed when f is strictly convex
- Nelder-Mead is globally convergent when $n = 1$
- Simplex may become flat or needle shaped

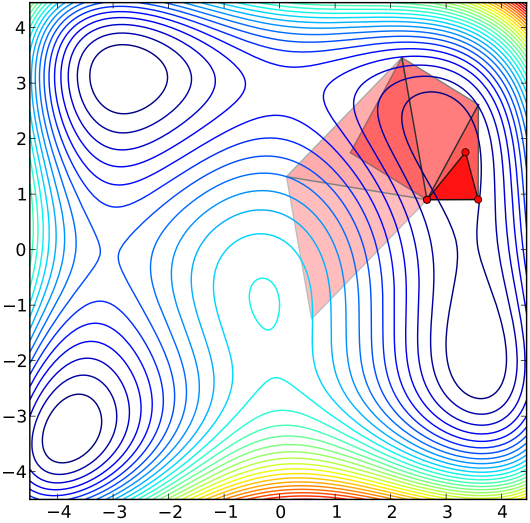
Improved variant

- Adds safeguard rotation if normalized volume deteriorates too much
- Imposes sufficient decrease condition
- Globally convergent to stationary point if f is continuously differentiable

Matlab implementation: `fminsearch`

Example: Nelder-Mead method

Himmelblau's function: $f(x_1, x_2) = (x_1^2 + x_2 - 11)^2 + (x_1 + x_2^2 - 7)^2$



Trust-region methods for DFO (Model-based methods)

Trust-region methods for DFO typically:

- attempt to form quadratic model of the objective function by interpolation/regression.

$$m_k(\mathbf{x}_k + \mathbf{p}) = f(\mathbf{x}_k) + \mathbf{g}_k^T \mathbf{p} + \frac{1}{2}\mathbf{p}^T H_k \mathbf{p}$$

- calculate the step $\mathbf{p}_k$ by solving

$$\min_{\mathbf{p}} \; m_k(\mathbf{x}_k + \mathbf{p}), \qquad \text{subject to } \|\mathbf{p}\|_2 \leq \Delta. \tag{1}$$

- set $\mathbf{x}_{k+1}$ to $\mathbf{x}_k + \mathbf{p}$ (success) or to $\mathbf{x}_k$ (unsuccess) and update Δ_k depending on the value of

$$\rho_k = \frac{f(\mathbf{x}_k) - f(\mathbf{x}_k + \mathbf{p}_k)}{m_k(\mathbf{x}_k) - m(\mathbf{x}_k + \mathbf{p}_k)}.$$

Outline of Trust-region methods for DFO

Algorithm

Choose an interpolation set $Y = \{\mathbf{y}_1, \mathbf{y}_2, \cdots, \mathbf{y}_q\}$ and $\mathbf{x}_0$ such that $f(\mathbf{x}_0) \leq f(\mathbf{y}_i)$ for all $\mathbf{y}_i \in Y$.
Set Δ_0 and $\eta \in (0, 1)$.
loop
    Form the model m_k by interpolation;
    Compute $\mathbf{p}_k$ by (approximately) solving (1);
    $\mathbf{x}_{new} = \mathbf{x}_k + \mathbf{p}_k$;
    $\rho_k = (f(\mathbf{x}_k) - f(\mathbf{x}_{new}))/(m_k(\mathbf{x}_k) - m_k(\mathbf{x}_{new}))$;

    if $\rho_k \geq \eta$ **then**
        Replace an element of Y by $\mathbf{x}_{new}$;
        Increase Δ_k, accept $\mathbf{x}_{new}$ and move to the next iteration;
    else if the set Y need be improved **then**
        Replace at least one element in Y;
        Keep Δ_k;
        Choose $\mathbf{x}_{new}$ as an element in Y with the lowest function value and recompute ρ_k;
        Accept or reject $\mathbf{x}_{new}$ according to ρ_k;
    else
        Reduce Δ_k, reject $\mathbf{x}_{new}$ and move to the next iteration;
    end if
end loop

Interpolation models

Linear model

$$m(\mathbf{x}_k + \mathbf{p}) = f(\mathbf{x}_k) + \mathbf{g}^T \mathbf{p}$$

- Model parameters are $\mathbf{g} \in \mathbb{R}^n$, i.e., n parameters.

Interpolation conditions

$$m(\mathbf{y}_i) = f(\mathbf{y}_i) \iff (\mathbf{y}_i - \mathbf{x}_k)^T \mathbf{g} = f(\mathbf{y}_i) - f(\mathbf{x}_k), \quad i = 1, 2, \ldots, q$$

The model is unique if $q = n$ and the system is nonsingular.

- cost of computing model from scratch is $O(n^3)$
- cost of updating model is $O(n^2)$

Interpolation models

Quadratic model

$$m(\mathbf{x}_k + \mathbf{p}) = f(\mathbf{x}_k) + \mathbf{g}^T\mathbf{p} + \tfrac{1}{2}\mathbf{p}^T H\mathbf{p}$$
$$= f(\mathbf{x}_k) + \mathbf{g}^T\mathbf{p} + \sum_{i<j} H_{ij}p_ip_j + \tfrac{1}{2}\sum_i H_{ii}p_i^2 = f(\mathbf{x}_k) + \hat{\mathbf{g}}^T\hat{\mathbf{p}}$$

- $\hat{\mathbf{p}} = (\mathbf{p}^T, \{p_ip_j\}_{i<j}, \{p_i^2/\sqrt{2}\})^T$.
- $\hat{\mathbf{g}} = (\mathbf{g}^T, \{H_{ij}\}_{i<j}, \{H_{ii}/\sqrt{2}\})^T$.
- Model parameters are $\mathbf{g} \in \mathbb{R}^n$ and symmetric matrix $H \in \mathbb{R}^{n\times n}$, i.e., $\frac{1}{2}n(n+3)$ parameters.

Interpolation conditions

$$m(\mathbf{y}_i) = f(\mathbf{y}_i) \quad i = 1, 2, \ldots, q$$

The model is unique if $q = \frac{1}{2}n(n+3)$ and the system is nonsingular.

- cost of computing model from scratch is $O(n^6)$
- cost of updating model is $O(n^4)$

Polynomial models

Given the interpolation points $\{\mathbf{y}_1, \cdots, \mathbf{y}_q\}$ and a polynomial basis $\{\phi_i(\cdot)\}_{i=1}^q$, one considers a system of linear equations:

$$m_k(\mathbf{y}_j) = \sum_{i=1}^{q} \alpha_i \phi_i(\mathbf{y}_j) = f(\mathbf{y}_j) \qquad j = 1, \cdots, q$$

for some coefficients α_i. It can be written as

$$M\boldsymbol{\alpha} = \mathbf{b}$$

where

$$M = \begin{bmatrix} \phi_1(\mathbf{y}_1) & \cdots & \phi_q(\mathbf{y}_1) \\ \vdots & \vdots & \vdots \\ \phi_1(\mathbf{y}_q) & \cdots & \phi_q(\mathbf{y}_q) \end{bmatrix} \qquad \mathbf{b} = \begin{bmatrix} f(\mathbf{y}_1) \\ \vdots \\ f(\mathbf{y}_q) \end{bmatrix}$$

Example: $\phi = \{1, x_1, x_2, x_1^2/2, x_2^2/2, x_1 x_2\}$.

Simplex gradient

The gradient $\mathbf{g}$ of linear interpolation model

$$m(\mathbf{x}) = f(\mathbf{y}_0) + (\mathbf{x} - \mathbf{y}_0)^T \mathbf{g}, \quad \mathbf{x}, \mathbf{y}_0 \in \mathbb{R}^n$$

- based on $n+1$ affinely independent points $\mathbf{y}_0, \mathbf{y}_1, \ldots, \mathbf{y}_n$

$$f(\mathbf{y}_i) = f(\mathbf{y}_0) + (\mathbf{y}_i - \mathbf{y}_0)^T \mathbf{g}, \quad i = 1, \ldots, n$$

- based on $q > n+1$ points: $\mathbf{g}$ is the solution to least-squares problem

$$\min_{\mathbf{g}} \sum_{i=1}^{q} (f(\mathbf{y}_i) - f(\mathbf{y}_0) - (\mathbf{y}_i - \mathbf{y}_0)^T \mathbf{g})^2$$

Simplex gradient

Special cases

- Forward-difference approximation ($n+1$ points)

$$\mathbf{y}_0, \quad \mathbf{y}_i = \mathbf{y}_0 + \epsilon \mathbf{e}_i, \quad i = 1, \ldots, n$$

- Central-difference approximation ($2n+1$ points)

$$\mathbf{y}_0, \quad \mathbf{y}_i = \mathbf{y}_0 + \epsilon \mathbf{e}_i,\ \mathbf{y}_{n+i} = \mathbf{y}_0 - \epsilon \mathbf{e}_i, \quad i = 1, \ldots, n$$

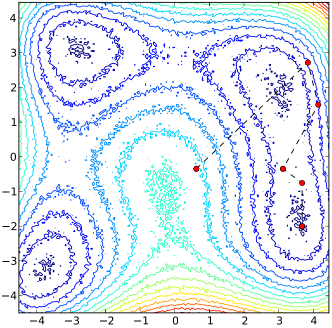
minimizes the least-squares objective

$$\sum_{i=1}^{n} \left[(f(\mathbf{y}_0 + \epsilon \mathbf{e}_i) - f(\mathbf{y}_0) - \epsilon \mathbf{g}_i)^2 + (f(\mathbf{y}_0 - \epsilon \mathbf{e}_i) - f(\mathbf{y}_0) + \epsilon \mathbf{g}_i)^2 \right]$$

Implicit-filtering methods

When objective function is

- stochastic or corrupted by noise
- nonsmooth
- not defined everywhere
- discontinuous



Implicit-filtering method is a variant of the steepest descent method with line search.

- It uses the simplex gradient.
- The simplex gradient may be improved by applying a quasi-Newton update.
- It performs a line search along the negative computed direction.

Matlab implementation: `http://www.siam.org/books/se23/`