Performance Estimation of Optimization Methods

An overview

François Glineur

UCLouvain

CORE (Center for Operations Research and Econometrics)
ICTEAM (Information and Communication Technologies, Electronics and Applied Mathematics Institute)

Joint work with

Nizar Bousselmi, Etienne de Klerk, Julien Hendrickx, Yassine Kamri, Panos Patrinos, Teodor Rotaru, Adrien Taylor, Pierre Vernimmen, Moslem Zamani

Funding: EU MSCA, FNRS/FRIA, KULeuven GlobalPartnership, UCLouvain FSR

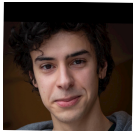
Dutch Day of Optimization 2026

Groningen – November 12, 2026



UCLouvain
Institute for Information and Communication Technologies, Electronics and Applied Mathematics

Research is teamwork!



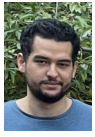
Nizar Bousselmi



Etienne de Klerk



Julien Hendrickx



Yassine Kamri



Panos Patrinos



Teodor Rotaru



Adrien Taylor

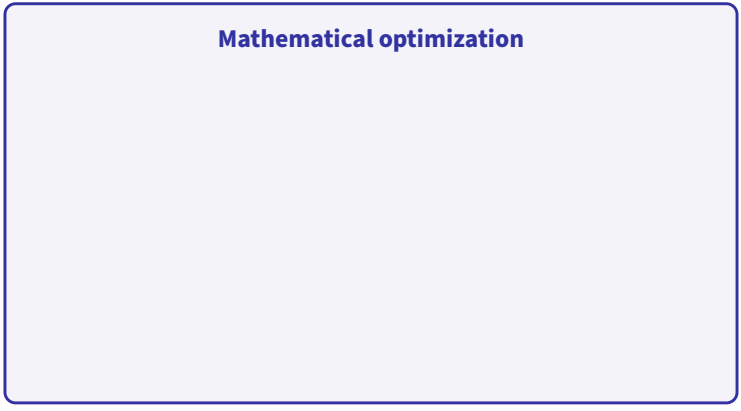


Pierre Vernimmen

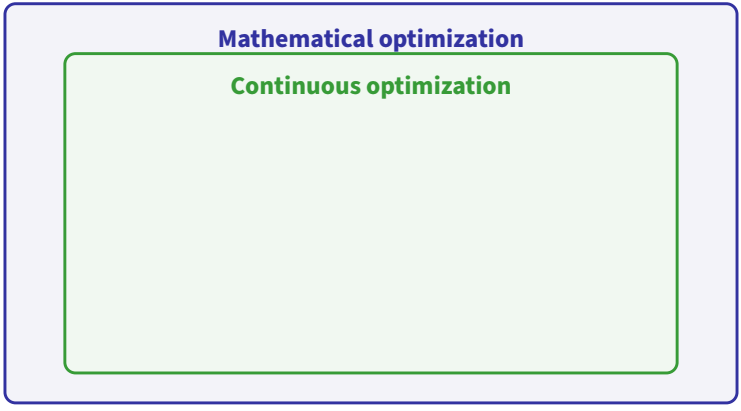


Moslem Zamani

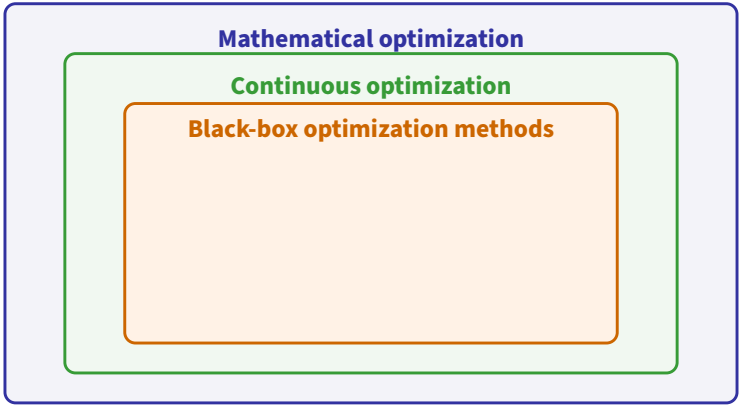
Context: where this talk sits



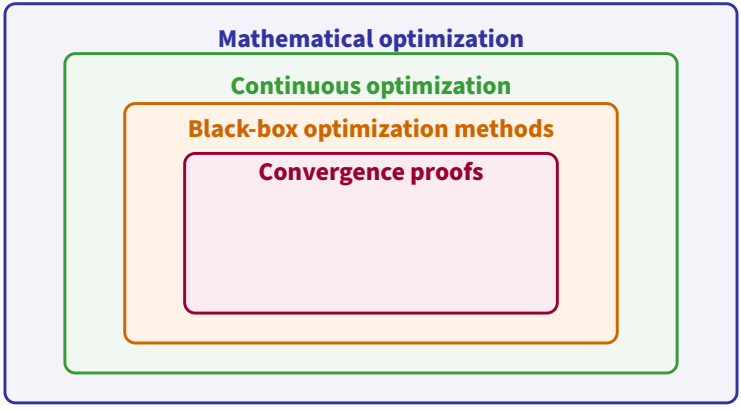
Context: where this talk sits



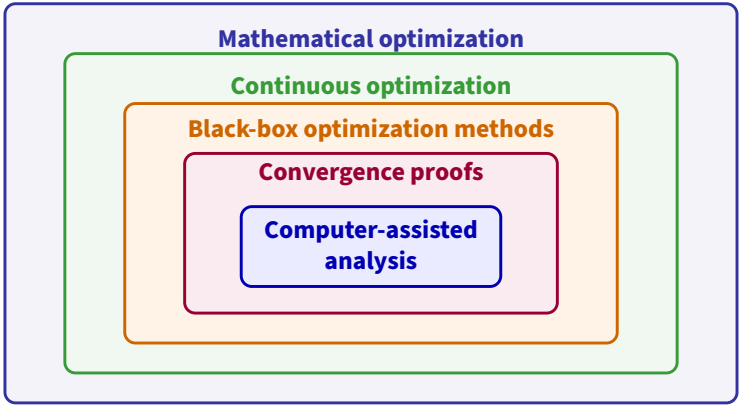
Context: where this talk sits



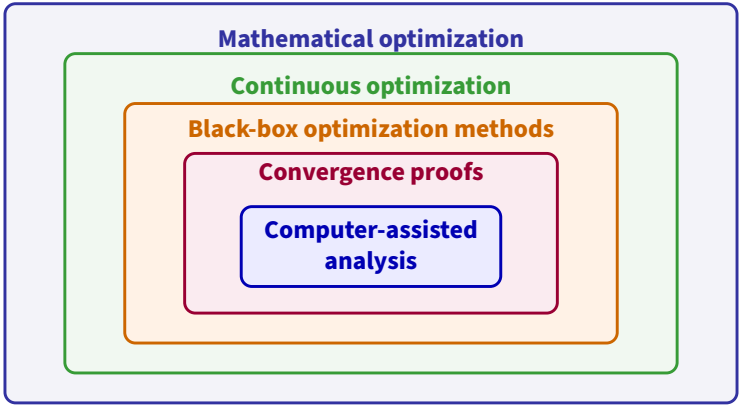
Context: where this talk sits



Context: where this talk sits



Context: where this talk sits



Our topic today:

Performance Estimation Problem = PEP

A methodology for computer-assisted convergence analysis of black-box optimization methods

Outline

Convergence bounds in optimization

What is Performance Estimation?

How does PEP work?

A selection of PEP results

Open questions

Conclusions References

Outline

Convergence bounds in optimization

What is Performance Estimation?

How does PEP work?

Wooclap time

A selection of PEP results

Open questions

Conclusions

References

Black-box first-order optimization methods

Optimization problem

$$\min_{x \in \mathbb{R}^d} f(x)$$

- f belongs to some function class
- dimension d may be very large

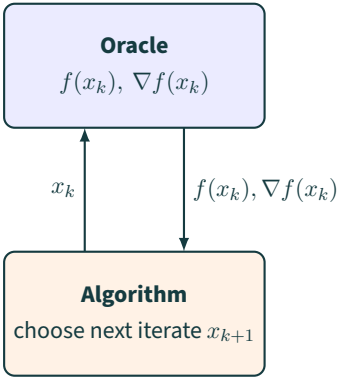
Black-box oracle model

Query x_k; receive $f_k = f(x_k)$ and $g_k = \nabla f(x_k)$

General first-order method

$$x_{k+1} = \mathcal{M}_k(x_0, f_0, g_0, f_1, g_1, \ldots, f_k, g_k)$$

Uses only the oracle history.



Goal

With $R = \|x_0 - x_*\|$:

$$f(x_N) - f_* \leq \frac{LR^2}{2N}$$

Examples of function classes

- **L-smooth nonconvex functions**

$$\|\nabla f(x) - \nabla f(y)\| \leq L\|x - y\| \quad \forall x, y$$

- **L-smooth convex functions**

$$f(y) \geq f(x) + \langle \nabla f(x), y - x\rangle, \quad \|\nabla f(x) - \nabla f(y)\| \leq L\|x - y\| \quad \forall x, y$$

- **μ-strongly convex functions**

$$f(y) \geq f(x) + \langle \nabla f(x), y - x\rangle + \frac{\mu}{2}\|y - x\|^2$$

  Equivalently, $x \mapsto f(x) - \frac{\mu}{2}\|x\|^2$ is convex.

- **Unified class $\mathcal{F}_{\mu,L}$:** curvature between μ and L, with $-\infty \leq \mu \leq L \leq +\infty$.

$$\mu \leq \lambda_k(\nabla^2 f(x)) \leq L \quad \forall k, x \qquad \text{when } f \in \mathcal{C}^2.$$

Examples of function classes

- **Convex functions with bounded subgradients**

$$f(y) \geq f(x) + \langle g_x, y - x \rangle \quad \forall x, y, \quad \forall g_x \in \partial f(x),$$

together with

$$\|g_x\| \leq B \quad \forall x, \quad \forall g_x \in \partial f(x).$$

which is equivalent to f being B-Lipschitz

$$|f(x) - f(y)| \leq B\|x - y\| \quad \forall x, y.$$

Classical examples of convergence results

(1) Smooth convex minimization

Assume f is convex with L-Lipschitz gradient.
Gradient method with stepsize $1/L$:

$$x_{k+1} = x_k - \frac{1}{L}\nabla f(x_k).$$

Then

$$f(x_N) - f_* \leq \frac{L\|x_0 - x_*\|^2}{2N}.$$

$$\boxed{f(x_N) - f_* = O(1/N)}$$

[Nesterov, 2018]

Classical examples of convergence results

(2) Smooth nonconvex minimization

Assume f has L-Lipschitz gradient.
Gradient method with stepsize $1/L$:

$$x_{k+1} = x_k - \frac{1}{L}\nabla f(x_k).$$

Then

$$\min_{0\leq k\leq N} \|\nabla f(x_k)\|^2 \leq \frac{2L(f(x_0) - f_*)}{N+1}.$$

$$\boxed{\min_k \|\nabla f(x_k)\|^2 = O(1/N)}$$

for example [Nesterov, 2018]

Classical examples of convergence results

(3) Accelerated gradient method

Assume f is convex with L-Lipschitz gradient.
Nesterov's accelerated fast gradient method [Nesterov, 1983]

$$\left\{ \begin{array}{rcl} y_k & = & x_k + \frac{k-1}{k+2}(x_k - x_{k-1}) \\ x_{k+1} & = & y_k - \frac{1}{L}\nabla f(y_k) \end{array} \right. \qquad \text{(with } x_{-1} = x_0\text{)}$$

$$f(x_N) - f_* \leq \frac{2L\|x_0 - x_*\|^2}{(N+1)^2} = O(1/N^2)$$

Lower complexity bound

The $O(1/N^2)$ rate is optimal for black-box first-order methods. [Nemirovsky–Yudin, 1983]

Classical examples of convergence results

(4) Subgradient method

Assume f is convex and B-Lipschitz, with $\|x_0 - x_*\| \leq R$.
Subgradient method with optimal constant stepsize $\gamma = R/(B\sqrt{N+1})$:

$$x_{k+1} = x_k - \gamma g_k, \qquad g_k \in \partial f(x_k), \qquad k = 0, \ldots, N-1.$$

The average iterate $\bar{x}_N = (N+1)^{-1} \sum_{k=0}^{N} x_k$ satisfies

$$f(\bar{x}_N) - f_* \leq \frac{BR}{\sqrt{N+1}}. \qquad \text{[Polyak, 1960s]}$$

$$\boxed{f(\bar{x}_N) - f_* = O(1/\sqrt{N})}$$

Lower complexity bound [Drori–Teboulle, 2016, Thm. A.1]

After N steps, any deterministic black-box first-order method has worst-case error at least $BR/\sqrt{N+1}$ in sufficiently large dimension.

A typical convergence proof: Nesterov accelerated gradient

[Ang, 2023]

NAGD converges rate $\mathcal{O}\left(\frac{1}{k^2}\right)$ proof 1/6 **Stage 1: make use of convexity & smoothness**

- f cvx: $(\forall \boldsymbol{x} \forall \boldsymbol{y})\Big\{f(\boldsymbol{y}) \geq f(\boldsymbol{x}) + \langle \nabla f(\boldsymbol{x}), \boldsymbol{y} - \boldsymbol{x}\rangle\Big\}$ gives

$$-f(\boldsymbol{y}) \leq -f(\boldsymbol{x}) + \langle \nabla f(\boldsymbol{x}), \boldsymbol{x} - \boldsymbol{y}\rangle \tag{5}$$

- f L-smooth $(\forall \boldsymbol{a} \forall \boldsymbol{b})\Big\{f(\boldsymbol{a}) - f(\boldsymbol{b}) \leq \langle \nabla f(\boldsymbol{b}), \boldsymbol{a} - \boldsymbol{b}\rangle + \frac{L}{2}\|\boldsymbol{a} - \boldsymbol{b}\|_2^2\Big\}$, with $\boldsymbol{a} = \boldsymbol{x} - \frac{1}{L}\nabla f(\boldsymbol{x}), \boldsymbol{b} = \boldsymbol{x}$,

$$f\Big(\boldsymbol{x} - \frac{1}{L}\nabla f(\boldsymbol{x})\Big) - f(\boldsymbol{x}) \;\leq\; -\frac{1}{L}\|\nabla f(\boldsymbol{x})\|_2^2 + \frac{1}{2L}\|\nabla f(\boldsymbol{x})\|_2^2 = \frac{-1}{2L}\|\nabla f(\boldsymbol{x})\|_2^2. \tag{6}$$

- (5) + (6) will cancel $-f(\boldsymbol{x})$ and give

$$f\Big(\boldsymbol{x} - \frac{1}{L}\nabla f(\boldsymbol{x})\Big) - f(\boldsymbol{y}) \;\leq\; \frac{-1}{2L}\|\nabla f(\boldsymbol{x})\|_2^2 + \langle \nabla f(\boldsymbol{x}), \boldsymbol{x} - \boldsymbol{y}\rangle. \tag{7}$$

- Put $\boldsymbol{x} = \boldsymbol{x}_k$, $\boldsymbol{y} = \boldsymbol{x}^*$ in (7)

$$f\Big(\boldsymbol{x}_k - \frac{1}{L}\nabla f(\boldsymbol{x}_k)\Big) - f^* \;\leq\; \frac{-1}{2L}\|\nabla f(\boldsymbol{x}_k)\|_2^2 + \langle \nabla f(\boldsymbol{x}_k), \boldsymbol{x}_k - \boldsymbol{x}^*\rangle. \tag{8}$$

- put $\boldsymbol{x} = \boldsymbol{x}_k$, $\boldsymbol{y} = \boldsymbol{y}_k$ in (7)

$$f\Big(\boldsymbol{x}_k - \frac{1}{L}\nabla f(\boldsymbol{x}_k)\Big) - f(\boldsymbol{y}_k) \;\leq\; \frac{-1}{2L}\|\nabla f(\boldsymbol{x}_k)\|_2^2 + \langle \nabla f(\boldsymbol{x}_k), \boldsymbol{x}_k - \boldsymbol{y}_k\rangle. \tag{9}$$

- Proof overview: (8), (9) link $f(\boldsymbol{y}_{k+1}), f(\boldsymbol{y}_k)$ and f^*. We see $\nabla f(\boldsymbol{x}_k)$ appear in (8), (9) but not in the convergence result, so we eliminate $\nabla f(\boldsymbol{x}_k)$ in (8), (9).

A typical convergence proof: Nesterov accelerated gradient

[Ang, 2023]

Proof 2/6 **Stage 2: eliminate gradient**

$$\boldsymbol{y}_{k+1} = \boldsymbol{x}_k - \frac{1}{L}\nabla f(\boldsymbol{x}_k) \tag{1}$$
$$f\Big(\boldsymbol{x}_k - \frac{1}{L}\nabla f(\boldsymbol{x}_k)\Big) - f^* \leq \frac{1}{2L}\|\nabla f(\boldsymbol{x}_k)\|_2^2 + \langle \nabla f(\boldsymbol{x}_k), \boldsymbol{x}_k - \boldsymbol{x}^*\rangle. \tag{8}$$
$$f\Big(\boldsymbol{x}_k - \frac{1}{L}\nabla f(\boldsymbol{x}_k)\Big) - f(\boldsymbol{y}_k) \leq \frac{1}{2L}\|\nabla f(\boldsymbol{x}_k)\|_2^2 + \langle \nabla f(\boldsymbol{x}_k), \boldsymbol{x}_k - \boldsymbol{y}_k\rangle. \tag{9}$$

- Simplify notation, let $\delta_k := f(\boldsymbol{y}_k) - f^*$, then

$$f\Big(\boldsymbol{x}_k - \frac{1}{L}\nabla f(\boldsymbol{x}_k)\Big) \overset{(1)}{=} f(\boldsymbol{y}_{k+1}) \tag{10}$$
$$f\Big(\boldsymbol{x}_k - \frac{1}{L}\nabla f(\boldsymbol{x}_k)\Big) - f^* \overset{(10),\delta_k}{=} \delta_{k+1} \tag{11}$$
$$\begin{aligned} f\Big(\boldsymbol{x}_k - \frac{1}{L}\nabla f(\boldsymbol{x}_k)\Big) - f(\boldsymbol{y}_k) &= f\Big(\boldsymbol{x}_k - \frac{1}{L}\nabla f(\boldsymbol{x}_k)\Big) - f^* - \Big(f(\boldsymbol{y}_k) - f^*\Big) \\ &= \delta_{k+1} - \delta_k \end{aligned} \tag{12}$$
$$\nabla f(\boldsymbol{x}_k) \overset{(1)}{=} -L(\boldsymbol{y}_{k+1} - \boldsymbol{x}_k) \tag{13}$$
$$\|\nabla f(\boldsymbol{x}_k)\|_2^2 \overset{(13)}{=} L^2\|\boldsymbol{y}_{k+1} - \boldsymbol{x}_k\|_2^2 \tag{14}$$

- Put (11,13,14) into (8)

$$\delta_{k+1} \leq -\frac{L}{2}\|\boldsymbol{y}_{k+1} - \boldsymbol{x}_k\|_2^2 - L\langle \boldsymbol{y}_{k+1} - \boldsymbol{x}_k,\, \boldsymbol{x}_k - \boldsymbol{x}^*\rangle. \tag{15}$$

- Put (12,13,14) into (9)

$$\delta_{k+1} - \delta_k \leq -\frac{L}{2}\|\boldsymbol{y}_{k+1} - \boldsymbol{x}_k\|_2^2 - L\langle \boldsymbol{y}_{k+1} - \boldsymbol{x}_k,\, \boldsymbol{x}_k - \boldsymbol{y}_k\rangle. \tag{16}$$

5 / 10

A typical convergence proof: Nesterov accelerated gradient

[Ang, 2023]

$$\begin{array}{rcll} \lambda_k & = & \frac{1}{2}\left(1+\sqrt{1+4\lambda_{k-1}^2}\right) & (4) \\ \delta_{k+1} & \leq & -\frac{L}{2}\|\boldsymbol{y}_{k+1}-\boldsymbol{x}_k\|_2^2 - L\langle \boldsymbol{y}_{k+1}-\boldsymbol{x}_k,\, \boldsymbol{x}_k-\boldsymbol{x}^*\rangle & (15) \\ \delta_{k+1}-\delta_k & \leq & -\frac{L}{2}\|\boldsymbol{y}_{k+1}-\boldsymbol{x}_k\|_2^2 - L\langle \boldsymbol{y}_{k+1}-\boldsymbol{x}_k,\, \boldsymbol{x}_k-\boldsymbol{y}_k\rangle & (16) \end{array}$$

Proof 3/6 **Stage 3: form telescoping sum**

- **Tricky step**: consider (15) + $(\lambda_k - 1)$(16).

$$\text{Left-hand size of (15)} + (\lambda_k-1)(16) \;=\; \delta_{k+1} + (\lambda_k-1)(\delta_{k+1}-\delta_k) \;=\; \lambda_k\delta_{k+1} - (\lambda_k-1)\delta_k.$$

- Right-hand side of (15) + $(\lambda_k - 1)$(16)

$$\begin{aligned} & -\frac{L}{2}\|\boldsymbol{y}_{k+1}-\boldsymbol{x}_k\|_2^2 - L\Big\langle \boldsymbol{y}_{k+1}-\boldsymbol{x}_k\,,\, \boldsymbol{x}_k-\boldsymbol{x}^*\Big\rangle + (\lambda_k-1)\left(\frac{-L}{2}\|\boldsymbol{y}_{k+1}-\boldsymbol{x}_k\|_2^2 - L\Big\langle \boldsymbol{y}_{k+1}-\boldsymbol{x}_k\,,\, \boldsymbol{x}_k-\boldsymbol{y}_k\Big\rangle\right) \\ = \;& -\frac{\lambda_k L}{2}\|\boldsymbol{y}_{k+1}-\boldsymbol{x}_k\|_2^2 - L\Big\langle \boldsymbol{y}_{k+1}-\boldsymbol{x}_k\,,\, \boldsymbol{x}_k-\boldsymbol{x}^*+(\lambda_k-1)(\boldsymbol{x}_k-\boldsymbol{y}_k)\Big\rangle \\ = \;& -\frac{\lambda_k L}{2}\|\boldsymbol{y}_{k+1}-\boldsymbol{x}_k\|_2^2 - L\Big\langle \boldsymbol{y}_{k+1}-\boldsymbol{x}_k,\, \lambda_k\boldsymbol{x}_k-(\lambda_k-1)\boldsymbol{y}_k-\boldsymbol{x}^*\Big\rangle \end{aligned}$$

- By LHS = RHS $\lambda_k\delta_{k+1} - (\lambda_k-1)\delta_k \;\leq\; -\frac{\lambda_k L}{2}\|\boldsymbol{y}_{k+1}-\boldsymbol{x}_k\|_2^2 - L\Big\langle \boldsymbol{y}_{k+1}-\boldsymbol{x}_k,\, \lambda_k\boldsymbol{x}_k-(\lambda_k-1)\boldsymbol{y}_k-\boldsymbol{x}^*\Big\rangle$.
  Multiply the inequality with λ_k:

$$\begin{aligned} \lambda_k^2\delta_{k+1} - \lambda_k(\lambda_k-1)\delta_k \;&\leq\; -\frac{\lambda_k^2 L}{2}\|\boldsymbol{y}_{k+1}-\boldsymbol{x}_k\|_2^2 - \lambda_k L\Big\langle \boldsymbol{y}_{k+1}-\boldsymbol{x}_k,\, \lambda_k\boldsymbol{x}_k-(\lambda_k-1)\boldsymbol{y}_k-\boldsymbol{x}^*\Big\rangle \\ &=\; -\frac{L}{2}\left(\lambda_k^2\|\boldsymbol{y}_{k+1}-\boldsymbol{x}_k\|_2^2 + 2\lambda_k\Big\langle \boldsymbol{y}_{k+1}-\boldsymbol{x}_k,\, \lambda_k\boldsymbol{x}_k-(\lambda_k-1)\boldsymbol{y}_k-\boldsymbol{x}^*\Big\rangle\right). \quad (\#) \end{aligned}$$

- (4) gives $(2\lambda_k-1)^2 = 1+4\lambda_{k-1}^2 \iff 4\lambda_k^2 - 4\lambda_k + 1 = 1+4\lambda_{k-1}^2 \iff \lambda_{k-1}^2 = \lambda_k(\lambda_k-1)$, put this into (#) gives

$$\lambda_k^2\delta_{k+1} - \lambda_{k-1}^2\delta_k \leq -\frac{L}{2}\left(\lambda_k^2\|\boldsymbol{y}_{k+1}-\boldsymbol{x}_k\|_2^2 + 2\lambda_k\Big\langle \boldsymbol{y}_{k+1}-\boldsymbol{x}_k,\, \lambda_k\boldsymbol{x}_k-(\lambda_k-1)\boldsymbol{y}_k-\boldsymbol{x}^*\Big\rangle\right) \quad (17)$$

6 / 10

A typical convergence proof: Nesterov accelerated gradient

[Ang, 2023]

Proof 4/6

$$\lambda_k = \tfrac{1}{2}\left(1+\sqrt{1+4\lambda_{k-1}^2}\right) \tag{4}$$

$$\lambda_k^2\delta_{k+1} - \lambda_k(\lambda_k-1)\delta_k \leq -\frac{L}{2}\left(\lambda_k^2\|\boldsymbol{y}_{k+1}-\boldsymbol{x}_k\|_2^2 + 2\lambda_k\langle \boldsymbol{y}_{k+1}-\boldsymbol{x}_k,\ \lambda_k\boldsymbol{x}_k-(\lambda_k-1)\boldsymbol{y}_k-\boldsymbol{x}^*\rangle\right) \tag{17}$$

- Inspecting the inner product in (17) we see that it is completing squares (Thanks to Tony Silveti-Falls for figuring it out, 2023 Nov 3).

$$\|\lambda\boldsymbol{a}+\boldsymbol{b}\|_2^2 = \lambda^2\|\boldsymbol{a}\|_2^2 + 2\lambda\langle\boldsymbol{a},\boldsymbol{b}\rangle + \|\boldsymbol{b}\|_2^2 \iff \lambda^2\|\boldsymbol{a}\|_2^2 + 2\lambda\langle\boldsymbol{a},\boldsymbol{b}\rangle = \|\lambda\boldsymbol{a}+\boldsymbol{b}\|_2^2 - \|\boldsymbol{b}\|_2^2.$$

$$\begin{aligned}
&\lambda_k^2\|\boldsymbol{y}_{k+1}-\boldsymbol{x}_k\|_2^2 + 2\lambda_k\langle \boldsymbol{y}_{k+1}-\boldsymbol{x}_k,\ \lambda_k\boldsymbol{x}_k-(\lambda_k-1)\boldsymbol{y}_k-\boldsymbol{x}^*\rangle \\
= &\left\|\lambda(\boldsymbol{y}_{k+1}-\boldsymbol{x}_k)+\lambda_k\boldsymbol{x}_k-(\lambda_k-1)\boldsymbol{y}_k-\boldsymbol{x}^*\right\|_2^2 - \left\|\lambda_k\boldsymbol{x}_k-(\lambda_k-1)\boldsymbol{y}_k-\boldsymbol{x}^*\right\|_2^2 \\
= &\left\|\lambda_k\boldsymbol{y}_{k+1}-(\lambda_k-1)\boldsymbol{y}_k-\boldsymbol{x}^*\right\|_2^2 - \left\|\lambda_k\boldsymbol{x}_k-(\lambda_k-1)\boldsymbol{y}_k-\boldsymbol{x}^*\right\|_2^2.
\end{aligned}$$

- Using this (17) becomes

$$\lambda_k^2\delta_{k+1} - \lambda_{k-1}^2\delta_k \leq -\frac{L}{2}\left(\left\|\lambda_k\boldsymbol{y}_{k+1}-(\lambda_k-1)\boldsymbol{y}_k-\boldsymbol{x}^*\right\|_2^2 - \left\|\lambda_k\boldsymbol{x}_k-(\lambda_k-1)\boldsymbol{y}_k-\boldsymbol{x}^*\right\|_2^2\right). \tag{18}$$

- We have $\lambda_k\boldsymbol{x}_k-(\lambda_k-1)\boldsymbol{y}_k = (1-\lambda_{k-1})\boldsymbol{y}_{k-1}+\lambda_{k-1}\boldsymbol{y}_k$.

Proof: $\gamma_k \overset{(3)}{=} \frac{1-\lambda_k}{\lambda_{k+1}} \iff \gamma_k\lambda_{k+1} = 1-\lambda_k$.
By (2) $\boldsymbol{x}_{k+1} = (1-\gamma_k)\boldsymbol{y}_{k+1}+\gamma_k\boldsymbol{y}_k$ gives $x_{k+1} = \boldsymbol{y}_{k+1}+\gamma_k(\boldsymbol{y}_k-\boldsymbol{y}_{k+1})$, multiply with λ_{k+1} gives
$\lambda_{k+1}x_{k+1} = \lambda_{k+1}\boldsymbol{y}_{k+1}+\lambda_{k+1}\gamma_k(\boldsymbol{y}_k-\boldsymbol{y}_{k+1}) = \lambda_{k+1}\boldsymbol{y}_{k+1}+(1-\lambda_k)(\boldsymbol{y}_k-\boldsymbol{y}_{k+1})$, rearrange gives
$\lambda_{k+1}x_{k+1}-\lambda_{k+1}\boldsymbol{y}_{k+1} = (1-\lambda_k)(\boldsymbol{y}_k-\boldsymbol{y}_{k+1})$, add $\boldsymbol{y}_{k+1}$ on both side gives
$\lambda_{k+1}x_{k+1}-(\lambda_{k+1}-1)\boldsymbol{y}_{k+1} = (1-\lambda_k)\boldsymbol{y}_k+\lambda_k\boldsymbol{y}_{k+1}$. Move counter k by -1 gives the result.

So (18) becomes

$$\lambda_k^2\delta_{k+1} - \lambda_{k-1}^2\delta_k \leq -\frac{L}{2}\left(\left\|\lambda_k\boldsymbol{y}_{k+1}-(\lambda_k-1)\boldsymbol{y}_k-\boldsymbol{x}^*\right\|_2^2 - \left\|(1-\lambda_{k-1})\boldsymbol{y}_{k-1}+\lambda_{k-1}\boldsymbol{y}_k-\boldsymbol{x}^*\right\|_2^2\right).$$

7 / 10

A typical convergence proof: Nesterov accelerated gradient

[Ang, 2023]

Proof ... 5/6

We have $\lambda_k^2\delta_{k+1} - \lambda_{k-1}^2\delta_k \;\leq\; -\frac{L}{2}\left(\|\lambda_k \boldsymbol{y}_{k+1} - (\lambda_k - 1)\boldsymbol{y}_k - \boldsymbol{x}^*\|_2^2 - \|(1-\lambda_{k-1})\boldsymbol{y}_{k-1} + \lambda_{k-1}\boldsymbol{y}_k - \boldsymbol{x}^*\|_2^2\right)$.

Rearrange the second term to make the terms in right-hand side have similar form

$$\lambda_k^2\delta_{k+1} - \lambda_{k-1}^2\delta_k \;\leq\; -\frac{L}{2}\left(\|\lambda_k \boldsymbol{y}_{k+1} - (\lambda_k - 1)\boldsymbol{y}_k - \boldsymbol{x}^*\|_2^2 - \|\lambda_{k-1}\boldsymbol{y}_k - (\lambda_{k-1} - 1)\boldsymbol{y}_{k-1} - \boldsymbol{x}^*\|_2^2\right). \tag{19}$$

Let $\boldsymbol{u}_k = \lambda_k \boldsymbol{y}_{k+1} - (\lambda_k - 1)\boldsymbol{y}_k - \boldsymbol{x}^*$ so $\lambda_{k-1}\boldsymbol{y}_k - (\lambda_{k-1} - 1)\boldsymbol{y}_{k-1} - \boldsymbol{x}^* = \boldsymbol{u}_{k-1}$ and (19) becomes

$$\begin{array}{lll}
\lambda_k^2\delta_{k+1} - \lambda_{k-1}^2\delta_k & \leq & -\frac{L}{2}\left(\|\boldsymbol{u}_k\|_2^2 - \|\boldsymbol{u}_{k-1}\|_2^2\right) \\
\lambda_1^2\delta_2 - \lambda_0^2\delta_1 & \leq & -\frac{L}{2}\left(\|\boldsymbol{u}_1\|_2^2 - \|\boldsymbol{u}_0\|_2^2\right) \qquad \text{case } k = 1 \\
\lambda_2^2\delta_3 - \lambda_1^2\delta_2 & \leq & -\frac{L}{2}\left(\|\boldsymbol{u}_2\|_2^2 - \|\boldsymbol{u}_1\|_2^2\right) \qquad \text{case } k = 2 \\
& \vdots & \\
\lambda_{K-1}^2\delta_K - \lambda_{K-2}^2\delta_{K-1} & \leq & -\frac{L}{2}\left(\|\boldsymbol{u}_{K-1}\|_2^2 - \|\boldsymbol{u}_{K-2}\|_2^2\right) \qquad \text{case } k = K-1 \\
\lambda_{K-1}^2\delta_K - \lambda_0^2\delta_1 & \leq & -\frac{L}{2}\left(\|\boldsymbol{u}_{K-1}\|_2^2 - \|\boldsymbol{u}_0\|_2^2\right) \qquad \text{sum } k = 1 \text{ to } k = K-1 \\
& = & \frac{L}{2}\left(\|\boldsymbol{u}_0\|_2^2 - \|\boldsymbol{u}_{K-1}\|_2^2\right) \\
& \leq & \frac{L}{2}\|\boldsymbol{u}_0\|_2^2 \qquad\qquad \|\boldsymbol{u}_{K-1}\|_2^2 \geq 0
\end{array}$$

By definition, $\lambda_0 = 0$, $\boldsymbol{y}_0 = \boldsymbol{x}_0$, $\boldsymbol{u}_0 = \lambda_0\boldsymbol{y}_1 - (\lambda_0 - 1)\boldsymbol{y}_0 - \boldsymbol{x}^* \overset{\lambda_0=0}{=} \boldsymbol{y}_0 - \boldsymbol{x}^* \overset{\boldsymbol{y}_0=\boldsymbol{x}_0}{=} \boldsymbol{x}_0 - \boldsymbol{x}^*$, thus

$$\lambda_{K-1}^2\delta_K \leq \frac{L}{2}\|\boldsymbol{x}_0 - \boldsymbol{x}^*\|_2^2 \quad\Longrightarrow\quad \delta_K \leq \frac{L\|\boldsymbol{x}_0 - \boldsymbol{x}^*\|_2^2}{2\lambda_{K-1}^2}.$$

8 / 10

A typical convergence proof: Nesterov accelerated gradient

[Ang, 2023]

Proof ... 6/6

Lemma. $\lambda_{k-1} \geq \frac{k}{2}$.
Proof (by induction)

- Case $k = 0$ and $\lambda_0 = 0$. It is trivial $0 \geq 0/2$.
- Case $k = 1$. By definition,
  $\lambda_k = \frac{1+\sqrt{1+4\lambda_{k-1}^2}}{2} = \frac{1+\sqrt{1+4\cdot 0^2}}{2} = 1 > \frac{1}{2} = \frac{k}{2}\big|_{k=1}$
- Induction hypothesis: assume $\lambda_{n-1} \geq \frac{n}{2}$.
- Case $k = n$

$$\begin{aligned}
\lambda_n &= \frac{1+\sqrt{1+4\lambda_{n-1}^2}}{2} \\
&\geq \frac{1+\sqrt{1+4\left(\frac{n}{2}\right)^2}}{2} && \text{[Induction hypothesis]} \\
&= \frac{1+\sqrt{1+n^2}}{2} \\
&> \frac{1+\sqrt{n^2}}{2} \\
&= \frac{1+n}{2}. \quad \square
\end{aligned}$$

With $\lambda_{k-1} \geq \frac{k}{2}$, so

$$\frac{1}{\lambda_{k-1}^2} \leq \frac{4}{k^2}.$$

Therefore $\delta_K \leq \frac{L\|\boldsymbol{x}_0 - \boldsymbol{x}^*\|_2^2}{2\lambda_{K-1}^2}$ becomes

$$f(\boldsymbol{y}_K) - f^* \leq \frac{2L\|\boldsymbol{x}_0 - \boldsymbol{x}^*\|_2^2}{K^2}.$$

where $f(\boldsymbol{y}_K) - f^* =: \delta_K$. $\quad \square$

Rename K as k gives

$$f(\boldsymbol{y}_k) - f^* \leq \frac{2L\|x_0 - \boldsymbol{x}^*\|_2^2}{k^2}.$$

This { complicated / highly-involved / non-intuitive } proof is now completed.

9 / 10

Outline

Convergence bounds in optimization

What is Performance Estimation?

How does PEP work?

A selection of PEP results

Open questions

Conclusions References

Performance estimation of an optimization method

Goal of a PEP (Performance Estimation Problem) [Drori–Teboulle, 2014]:
compute the exact worst-case behavior

Example:

Performance estimation of an optimization method

Goal of a PEP (Performance Estimation Problem) [Drori–Teboulle, 2014]:
compute the exact worst-case behavior

- of a given optimization black-box method
  (considering a fixed number of iterations)

Example: after computing N steps of the gradient method with fixed unit step-size
$x_{k+1} = x_k - \nabla f(x_k)$

Performance estimation of an optimization method

Goal of a PEP (Performance Estimation Problem) [Drori–Teboulle, 2014]:
compute the exact worst-case behavior

- of a given optimization black-box method
  (considering a fixed number of iterations)
- applied to any function belonging to a given class
  (e.g. convex functions, possibly with some additional regularity property such as smoothness)

Example: after computing N steps of the gradient method with fixed unit step-size $x_{k+1} = x_k - \nabla f(x_k)$ applied to a convex function f with a 1-Lipschitz gradient and minimizer x^*

Performance estimation of an optimization method

Goal of a PEP (Performance Estimation Problem) [Drori–Teboulle, 2014]:
compute the exact worst-case behavior

- of a given optimization black-box method
  (considering a fixed number of iterations)
- applied to any function belonging to a given class
  (e.g. convex functions, possibly with some additional regularity property such as smoothness)
- from any starting point
  (satisfying some condition w.r.t. a minimizer)

Example: after computing N steps of the gradient method with fixed unit step-size $x_{k+1} = x_k - \nabla f(x_k)$ applied to a convex function f with a 1-Lipschitz gradient and minimizer x^* from an initial iterate satisfying $\|x_0 - x_*\| \leq 1$,

Performance estimation of an optimization method

Goal of a PEP (Performance Estimation Problem) [Drori–Teboulle, 2014]:
compute the exact worst-case behavior

- of a given optimization black-box method
  (considering a fixed number of iterations)
- applied to any function belonging to a given class
  (e.g. convex functions, possibly with some additional regularity property such as smoothness)
- from any starting point
  (satisfying some condition w.r.t. a minimizer)
- for a given performance criterion
  (e.g. objective accuracy, distance to solution, gradient norm)

Example: after computing N steps of the gradient method with fixed unit step-size $x_{k+1} = x_k - \nabla f(x_k)$ applied to a convex function f with a 1-Lipschitz gradient and minimizer x^* from an initial iterate satisfying $\|x_0 - x_*\| \leq 1$, what is the worst (largest) possible objective function accuracy for the last iterate $f(x_N) - f(x_*)$?

Output of a performance estimation problem

For a given PEP (Performance Estimation Problem) we will

Output of a performance estimation problem

For a given PEP (Performance Estimation Problem) we will

- compute the exact value of the performance criterion's worst-case = optimal value of PEP problem

Output of a performance estimation problem

For a given PEP (Performance Estimation Problem) we will

- compute the exact value of the performance criterion's worst-case = optimal value of PEP problem
- identify an explicit function (and starting point) achieving this worst-case value = primal solution of PEP problem + interpolation

Output of a performance estimation problem

For a given PEP (Performance Estimation Problem) we will

- compute the exact value of the performance criterion's worst-case = optimal value of PEP problem
- identify an explicit function (and starting point) achieving this worst-case value = primal solution of PEP problem + interpolation
- obtain an independently-checkable proof that this worst-case value is a valid (upper) bound on the performance criterion = dual multiplier of PEP problem

Output of a performance estimation problem

For a given PEP (Performance Estimation Problem) we will

- compute the exact value of the performance criterion's worst-case = optimal value of PEP problem
- identify an explicit function (and starting point) achieving this worst-case value = primal solution of PEP problem + interpolation
- obtain an independently-checkable proof that this worst-case value is a valid (upper) bound on the performance criterion = dual multiplier of PEP problem
- all three steps can be done either numerically or analytically

Output of a performance estimation problem

For a given PEP (Performance Estimation Problem) we will

- compute the exact value of the performance criterion's worst-case = optimal value of PEP problem
- identify an explicit function (and starting point) achieving this worst-case value = primal solution of PEP problem + interpolation
- obtain an independently-checkable proof that this worst-case value is a valid (upper) bound on the performance criterion = dual multiplier of PEP problem
- all three steps can be done either numerically or analytically

For a large class of first-order methods applied to convex composite optimization problems, these can be computed exactly using a semidefinite programming (SDP) problem.
[Taylor–Hendrickx–G, 2017b]

What you obtain when you solve a PEP

1. In all cases:
   numerical value for worst-case performance
   numerical description of worst-case function (interpolate primal)
   numerical proof of worst-case performance (dual multipliers)

What you obtain when you solve a PEP

1. In all cases:
   numerical value for worst-case performance
   numerical description of worst-case function (interpolate primal)
   numerical proof of worst-case performance (dual multipliers)

2. With extra work: explicit formulas for some/all of above
   $\rightarrow$ this requires fitting/guessing algebraic expressions in N

What you obtain when you solve a PEP

1. In all cases:
   numerical value for worst-case performance
   numerical description of worst-case function (interpolate primal)
   numerical proof of worst-case performance (dual multipliers)

2. With extra work: explicit formulas for some/all of above
   $\rightarrow$ this requires fitting/guessing algebraic expressions in N

3. In the best case: a rigorous mathematical proof
   $\rightarrow$ this requires a proof with (often tedious) reformulation of worst-case rate as inequality involving sums-of-squares

   Note: by theorem, all valid convergence bounds must be writable as sum-of-squares inequalities based on combinations of necessary and sufficient interpolation inequalities
   (hence any use of another inequality, while also leading to an upper bound on rate, may destroy its tightness!)

Very brief history of performance estimation

- Concept of performance estimation introduced in 2012 [Drori–Teboulle, 2014]: a relaxation gives bounds that are tight for some algorithms (using matching worst-case functions)

Very brief history of performance estimation

- Concept of performance estimation introduced in 2012 [Drori–Teboulle, 2014]: a relaxation gives bounds that are tight for some algorithms (using matching worst-case functions)
- Exact SDP formulation (2015 preprint) [Taylor–Hendrickx–G, 2017a], using necessary and sufficient interpolation conditions

Very brief history of performance estimation

- Concept of performance estimation introduced in 2012 [Drori–Teboulle, 2014]: a relaxation gives bounds that are tight for some algorithms (using matching worst-case functions)
- Exact SDP formulation (2015 preprint) [Taylor–Hendrickx–G, 2017a], using necessary and sufficient interpolation conditions
- Other approaches similar in spirit:
  integral quadratic constraints [Lessard et al., 2016]; Lyapunov/potential functions [Taylor–Bach, 2019]

Very brief history of performance estimation

- Concept of performance estimation introduced in 2012 [Drori–Teboulle, 2014]: a relaxation gives bounds that are tight for some algorithms (using matching worst-case functions)
- Exact SDP formulation (2015 preprint) [Taylor–Hendrickx–G, 2017a], using necessary and sufficient interpolation conditions
- Other approaches similar in spirit:
  integral quadratic constraints [Lessard et al., 2016]; Lyapunov/potential functions [Taylor–Bach, 2019]

- A growing subfield of algorithmic optimization (above three seminal papers total 1900+ citations)

Very brief history of performance estimation

- Concept of performance estimation introduced in 2012 [Drori–Teboulle, 2014]: a relaxation gives bounds that are tight for some algorithms (using matching worst-case functions)
- Exact SDP formulation (2015 preprint) [Taylor–Hendrickx–G, 2017a], using necessary and sufficient interpolation conditions
- Other approaches similar in spirit:
  integral quadratic constraints [Lessard et al., 2016]; Lyapunov/potential functions [Taylor–Bach, 2019]

- A growing subfield of algorithmic optimization (above three seminal papers total 1900+ citations)
- This talk mostly focuses on results obtained in UCLouvain, but there are many other recent exciting results

Outline

Convergence bounds in optimization

What is Performance Estimation?

How does PEP work?

A selection of PEP results

Open questions

Conclusions References

Obtaining an SDP formulation: a worked example

Exact SDP formulation for gradient descent with stepsize $1/L$. [Drori–Teboulle, 2014] [Taylor–Hendrickx–G, 2017a]

What is the worst possible objective function accuracy after applying N steps of the gradient method with fixed step-size $\frac{1}{L}$ to a convex function f with L-Lipschitz gradient and minimizer x_*, from an initial iterate satisfying $\|x_0 - x_*\| \leq R$?

Obtaining an SDP formulation: a worked example

Exact SDP formulation for gradient descent with stepsize $1/L$. [Drori–Teboulle, 2014] [Taylor–Hendrickx–G, 2017a]

What is the worst possible objective function accuracy after applying N steps of the gradient method with fixed step-size $\frac{1}{L}$ to a convex function f with L-Lipschitz gradient and minimizer x_*, from an initial iterate satisfying $\|x_0 - x_*\| \leq R$?

Formally, this is an infinite-dimensional problem over functions f:

$$\max_{f, x_*, x_0, x_1, \ldots, x_N} f(x_N) - f(x_*) \text{ s.t. } \begin{cases} f \text{ is proper and convex} \\ \text{f has an } L\text{-Lipschitz gradient} \\ x_* \text{ is a minimizer of } f \\ \|x_0 - x_*\| \leq R \\ x_{i+1} = x_i - \frac{1}{L} g_i \text{ for every } 0 \leq i < N \\ (\text{where } g_i = \nabla f(x_i) \text{ is the gradient}) \end{cases}$$

Step 1: use the black-box property

Performance estimation problem is infinite dimensional but the gradient method is black-box

(i.e. next iterate only depends on previously obtained oracle information, consisting of the function value and an (arbitrary) subgradient for each of the previous iterates)

In general, starting from initial x_0, a black-box method computes

$$\begin{aligned} x_1 &= \mathcal{M}_1\left(x_0, \mathcal{O}_f(x_0)\right), \\ x_2 &= \mathcal{M}_2\left(x_0, \mathcal{O}_f(x_0), \mathcal{O}_f(x_1)\right), \\ &\vdots \\ x_N &= \mathcal{M}_N\left(x_0, \mathcal{O}_f(x_0), \ldots, \mathcal{O}_f(x_{N-1})\right). \end{aligned}$$

Only depends on x_0 and the finite list of outputs from the oracle
($\mathcal{O}_f(x_i) = \{f(x_i), \nabla f(x_i)\}$)

Step 1: finite-dimensional reformulation using black-box

Hence infinite-dimensional variable f can be replaced by the list of oracle outputs, each defining a variable

$$f_i = f(x_i) \text{ and } g_i = \nabla f(x_i) \text{ for all } i = 0, 1, \dots N \text{ as well as } i = *$$

and performance estimation problem becomes

$$\left\{ \begin{array}{c} \max \\ x_*, x_0, x_1, \dots, x_N \\ f_*, f_0, f_1, \dots, f_N \\ g_*, g_0, g_1, \dots, g_N \end{array} \right. \quad f_N - f_* \text{ s.t.} \left\{ \begin{array}{l} \text{there exists proper and convex } f \text{ with} \\ \text{an } L\text{-Lipschitz gradient such that} \\ \qquad f(x_i) = f_i \text{ and } g_i = \nabla f(x_i) \\ \text{for all } i \in \{*, 0, 1, \dots N\} \\ g_* = 0 \\ \|x_0 - x_*\| \leq R \\ x_{i+1} = x_i - \frac{1}{L} g_i \text{ for every } 0 \leq i < N \end{array} \right.$$

which is finite-dimensional (minimizer condition became $g_* = 0$).
Remaining issue: imposing existence of suitable f compatible with f_i and g_i requires necessary and sufficient interpolating conditions

Step 2: introduce interpolating conditions

We need to express in our problem the fact that

> There exists proper and convex f with an L-Lipschitz gradient such that $f(x_i) = f_i$ and $g_i = \nabla f(x_i)$ for all $i \in I = \{*, 0, 1, \dots N\}$

i.e. given values of x_i, f_i and g_i we must guarantee existence of f

$\rightarrow$ we need necessary and sufficient conditions for interpolation

We use the following equivalence [Taylor–Hendrickx–G, 2017a]

there exists a convex f with L-Lipschitz gradient satisfying
$f(x_i) = f_i$ and $g_i = \nabla f(x_i)$ for every $i \in I$

$$\Leftrightarrow$$

$f_j \geq f_i + g_i^T(x_j - x_i) + \frac{1}{2L}\|g_i - g_j\|^2$ for every pair $i \in I, j \in I$

(can be extended to deal with the smooth strongly convex case, and the smooth nonconvex/weakly convex cases)

Necessary and sufficient conditions for convex interpolation

Example with a simpler class: convex functions [Taylor–Hendrickx–G, 2017a]

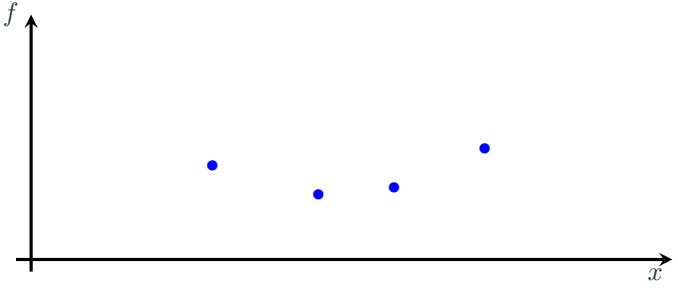
When is set $\{(x_i, g_i, f_i)\}_{i \in S}$ interpolable by a convex function?



Necessary and sufficient conditions for convex interpolation

Example with a simpler class: convex functions [Taylor–Hendrickx–G, 2017a]

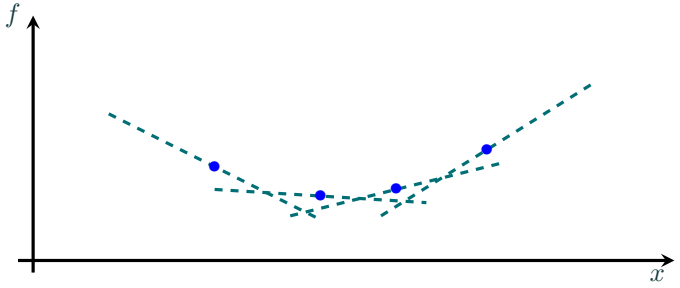
When is set $\{(x_i, g_i, f_i)\}_{i \in S}$ interpolable by a convex function?



Necessary and sufficient conditions for convex interpolation

Example with a simpler class: convex functions [Taylor–Hendrickx–G, 2017a]

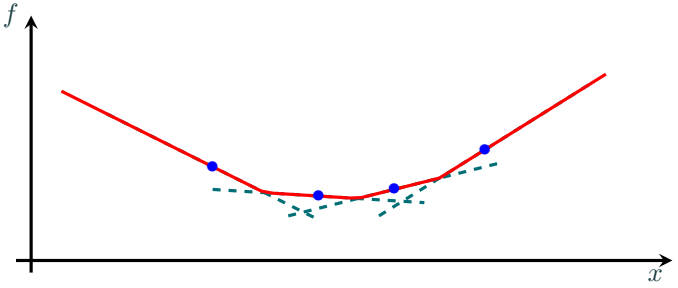
When is set $\{(x_i, g_i, f_i)\}_{i \in S}$ interpolable by a convex function?



Necessary and sufficient conditions for convex interpolation

Example with a simpler class: convex functions [Taylor–Hendrickx–G, 2017a]

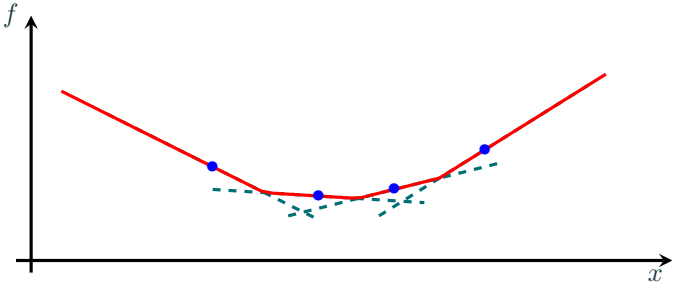
When is set $\{(x_i, g_i, f_i)\}_{i \in S}$ interpolable by a convex function?



Necessary and sufficient conditions for convex interpolation

Example with a simpler class: convex functions [Taylor–Hendrickx–G, 2017a]

When is set $\{(x_i, g_i, f_i)\}_{i \in S}$ interpolable by a convex function?

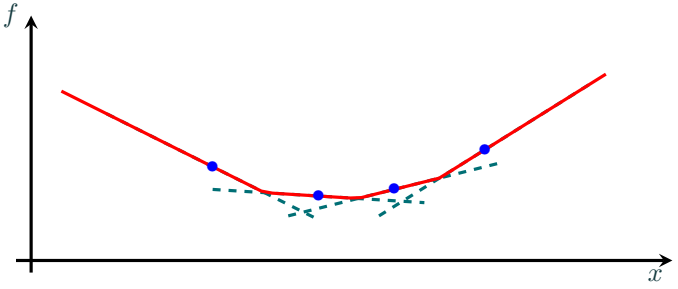


Subgradient inequalities $f_i \geq f_j + g_j^T (x_i - x_j)\ \forall i, j \in S$ are necessary

Necessary and sufficient conditions for convex interpolation

Example with a simpler class: convex functions [Taylor–Hendrickx–G, 2017a]

When is set $\{(x_i, g_i, f_i)\}_{i \in S}$ interpolable by a convex function?



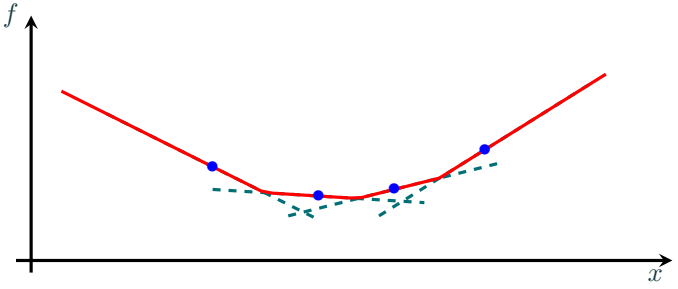
Subgradient inequalities $f_i \geq f_j + g_j^T (x_i - x_j)\ \forall i, j \in S$ are necessary

Function $f(x) = \max_j \left\{ f_j + g_j^T (x - x_j) \right\}$ is then a (piecewise linear) interpolating function (not unique!)

Necessary and sufficient conditions for convex interpolation

Example with a simpler class: convex functions [Taylor–Hendrickx–G, 2017a]

When is set $\{(x_i, g_i, f_i)\}_{i \in S}$ interpolable by a convex function?



Subgradient inequalities $f_i \geq f_j + g_j^T (x_i - x_j)\ \forall i, j \in S$ are necessary

Function $f(x) = \max_j \left\{ f_j + g_j^T (x - x_j) \right\}$ is then a (piecewise linear) interpolating function (not unique!)

Hence subgradient inequalities are necessary and sufficient for interpolation

Step 2 (cont.): introduce interpolating conditions

Performance estimation problem becomes
(with $I = \{*, 0, 1, \dots, N\}$)

$$\max_{\{x_i, f_i, g_i\}_{i \in I}} f_N - f_* \text{ s.t. } \begin{cases} f_j \geq f_i + g_i^T (x_j - x_i) + \frac{1}{2L} \|g_i - g_j\|^2 \, \forall i \neq j \in I \\ x_* = 0, f_* = 0, g_* = 0 \\ \|x_0 - x_*\| \leq R \\ x_{i+1} = x_i - \frac{1}{L} g_i \text{ for every } 0 \leq i < N \end{cases}$$

(note that constraints $f_* = 0$ and $x_* = 0$ can be added w.l.o.g.)

Worst-case is computed exactly with a finite explicit formulation ; idea first introduced in [Drori–Teboulle, 2014]

Unfortunately, problem is nonconvex (inner products $g_i^T x_j$)

Step 3: convexify using a Gram matrix

$$\max_{\{x_i,f_i,g_i\}_{i\in I}} f_N - f_* \text{ s.t. } \begin{cases} f_j \geq f_i + g_i^T(x_j - x_i) + \frac{1}{2L}\|g_i - g_j\|^2 \; \forall i \neq j \in I \\ x_* = 0, f_* = 0, g_* = 0 \\ \|x_0 - x_*\|^2 \leq R^2 \\ \left\|x_{i+1} - x_i + \frac{1}{L}g_i\right\|^2 = 0 \text{ for every } 0 \leq i < N \end{cases}$$

All terms are quadratic in x_i and g_i, and linear in f_i
Alternatively, you can (linearly) substitute out x_i for $i \geq 1$.
$\rightarrow$ introduce a Gram matrix for variables x_i and g_i !

Step 3: convexify using a Gram matrix

$$\max_{\{x_i,f_i,g_i\}_{i\in I}} f_N - f_* \text{ s.t. } \begin{cases} f_j \geq f_i + g_i^T(x_j - x_i) + \frac{1}{2L}\|g_i - g_j\|^2 \; \forall i \neq j \in I \\ x_* = 0, f_* = 0, g_* = 0 \\ \|x_0 - x_*\|^2 \leq R^2 \\ \left\|x_{i+1} - x_i + \frac{1}{L}g_i\right\|^2 = 0 \text{ for every } 0 \leq i < N \end{cases}$$

All terms are quadratic in x_i and g_i, and linear in f_i
Alternatively, you can (linearly) substitute out x_i for $i \geq 1$.
$\rightarrow$ introduce a Gram matrix for variables x_i and g_i !

Example $N = 1$: $\qquad P = (x_0 \mid x_1 \mid g_0 \mid g_1)$

$$G = P^T P = \begin{pmatrix} x_0^T x_0 & x_0^T x_1 & x_0^T g_0 & x_0^T g_1 \\ x_1^T x_0 & x_1^T x_1 & x_1^T g_0 & x_1^T g_1 \\ g_0^T x_0 & g_0^T x_1 & g_0^T g_0 & g_0^T g_1 \\ g_1^T x_0 & g_1^T x_1 & g_1^T g_0 & g_1^T g_1 \end{pmatrix} \succeq 0.$$

Step 3: convexify using a Gram matrix

For $N = 1$, use $P = (x_0 \mid x_1 \mid g_0 \mid g_1)$ and $G = P^\top P \in \mathbb{S}^4$, with $x_* = g_* = 0$ and $f_* = 0$.

$$
\begin{aligned}
\max_{f_0, f_1, G} \quad & f_1 \\
\text{s.t.} \quad & G \succeq 0, \qquad G_{11} \leq R^2, \\
& G_{11} + G_{22} - 2G_{12} + \frac{2}{L}(G_{23} - G_{13}) + \frac{G_{33}}{L^2} = 0, \\
& \frac{G_{33}}{2L} \leq f_0 \leq G_{13} - \frac{G_{33}}{2L}, \\
& \frac{G_{44}}{2L} \leq f_1 \leq G_{24} - \frac{G_{44}}{2L}, \\
& f_1 - f_0 \geq G_{23} - G_{13} + \frac{G_{33} + G_{44} - 2G_{34}}{2L}, \\
& f_0 - f_1 \geq G_{14} - G_{24} + \frac{G_{33} + G_{44} - 2G_{34}}{2L}.
\end{aligned}
$$

Linear in f_0, f_1 and G: a semidefinite optimization problem.

Step 4: recovering outputs

- **Value:** the SDP gives a worst-case upper bound, exact in sufficiently large dimension.

Step 4: recovering outputs

- **Value:** the SDP gives a worst-case upper bound, exact in sufficiently large dimension.
- **Proof:** dual multipliers combine interpolation inequalities into a certificate for the bound.

Step 4: recovering outputs

- **Value:** the SDP gives a worst-case upper bound, exact in sufficiently large dimension.
- **Proof:** dual multipliers combine interpolation inequalities into a certificate for the bound.
- **Worst-case instance:** factor an optimal Gram matrix

$$G = P^\top P, \qquad P = (x_0 \,|\, \cdots \,|\, x_N \,|\, g_0 \,|\, \cdots \,|\, g_N).$$

  The columns of P give the iterates and gradients; interpolate the data (x_i, g_i, f_i) to obtain a worst-case function.

- If $\operatorname{rank} G = r$, the bound is attained in every dimension $d \geq r$. Rank one gives tightness in every dimension. [Taylor–Hendrickx–G, 2017a]

Additional step: use homogeneity

Our performance estimation problem is parameterized by

- Lipschitz constant L for gradient
- Maximum distance R between starting point and a minimizer

Additional step: use homogeneity

Our performance estimation problem is parameterized by

- Lipschitz constant L for gradient
- Maximum distance R between starting point and a minimizer

To avoid having to solve for every value of L and R use homogeneity properties

- if $L \to \lambda L$ with $\lambda > 0$, worst-case is also multiplied by λ (proof: scale $f \to \lambda f$)
- if $R \to \lambda R$ with $\lambda > 0$, worst-case is also multiplied by λ^2 (proof: $f \to \lambda^2 f(\cdot/\lambda)$)

hence worst-case $w^*(L, R, N)$ is proportional to L and R^2

Additional step: use homogeneity

Our performance estimation problem is parameterized by

- Lipschitz constant L for gradient
- Maximum distance R between starting point and a minimizer

To avoid having to solve for every value of L and R use homogeneity properties

- if $L \to \lambda L$ with $\lambda > 0$, worst-case is also multiplied by λ (proof: scale $f \to \lambda f$)
- if $R \to \lambda R$ with $\lambda > 0$, worst-case is also multiplied by λ^2 (proof: $f \to \lambda^2 f(\cdot/\lambda)$)

hence worst-case $w^*(L, R, N)$ is proportional to L and R^2

$\to$ solve problem for $L = R = 1$, find worst-case value $w^*(1, 1, N)$ so that for general L and R we will have

$$f(x_N) - f(x_*) \leq w^*(L, R, N) = LR^2\, w^*(1, 1, N)$$

Only remaining parameter is N, number of steps

Example: Worst-case gradient method with constant $\frac{1}{L}$ stepsize

Apply PEP to N steps of gradient method with $\frac{1}{L}$ stepsize:

1. Notice numerical values of worst-case match formula

$$w^*(1,1,N) = \frac{1}{2(2N+1)}$$

(easy to see for small N, then verified for large N)

2. Guess from (factor of) Gram matrix the values of gradients g_i
Matrix is rank one, interpolating values leads to univariate piecewise linear-quadratic function f_ρ (Huber) with $\rho = \frac{1}{2N+1}$:

$$f_\rho(x) = \begin{cases} \rho|x| - \frac{\rho^2}{2} & \text{when } |x| \geq \rho \\ \frac{x^2}{2} & \text{when } |x| \leq \rho \end{cases}$$

Easy to check performance of method on f_ρ (starts with $x_0 = 1$, then $x_i = 1 - i\rho$), which establishes $w^* \geq \frac{1}{2(2N+1)}$

[Drori–Teboulle, 2014]

Example: Worst-case gradient method with constant $\frac{1}{L}$ stepsize

3. To prove upper bound on w^*, it is sufficient to exhibit a dual feasible solution with same value
   To describe a dual solution it is actually sufficient to give multipliers $\lambda_{i,j}$ for each interpolation inequality $i, j \in I$
   Again guess from numerical values ; e.g. for $N = 1$:

$$\lambda_{0,*} = \lambda_{1,*} = \lambda_{1,0} = \frac{1}{2} \quad \text{and all other multipliers zero}$$

4. Finally one needs to prove (analytically) that the above set of multipliers is feasible (hardest part is proving the slack matrix is positive semidefinite), leading finally to $w^* \leq \frac{1}{2(2N+1)}$

Example: Interpretation of proof with dual multipliers

One gradient step: $x_1 = x_0 - g_0/L$, with $g_i = \nabla f(x_i)$ and $r_0 = x_0 - x_*$.

Interpolation inequalities slacks [Taylor–Hendrickx–G, 2017a]

$$Q_{ij} := f_i - f_j - g_j^\top (x_i - x_j) - \frac{1}{2L}\|g_i - g_j\|^2 \geq 0.$$

Since $g_* = 0$, the following identity holds:

$$\begin{aligned} f_1 - f_* = \frac{L}{6}\|r_0\|^2 - \frac{1}{2}\big(Q_{01} + Q_{*0} + Q_{*1}\big) \\ - \frac{L}{6}\left\|r_0 - \frac{3}{2L}(g_0 + g_1)\right\|^2 - \frac{1}{8L}\|g_0 - g_1\|^2. \end{aligned}$$

Every subtracted term is nonnegative. Hence, if $\|r_0\| \leq R$,

$$\boxed{f_1 - f_* \leq \frac{LR^2}{6}.}$$

Software: from a PEP to code

PEPit: Performance Estimation in Python

[Goujaud, Moucer, G, Hendrickx, Taylor, Dieuleveut, 2024]

`pepit.readthedocs.io` Source code

Write the function class, algorithm and performance metric in Python. PEPit builds and solves the SDP, returning a bound and its certificate.

96 examples in 13 categories

- smooth, nonsmooth and composite optimization;
- stochastic, coordinate and adaptive methods;
- splitting, fixed points and monotone operators.

LLMs love Performance Estimation Problems! They can also code the SDP directly if you ask them nicely.

PEPit: Performance Estimation in Python

`pepit.readthedocs.io` Source code

Five gradient steps, $L = R = 1$.

```
from PEPit import PEP, functions
p = PEP()
f = p.declare_function(functions.SmoothConvexFunction, L=1)
xs, x = f.stationary_point(), p.set_initial_point()
p.set_initial_condition((x - xs)**2 <= 1)
for _ in range(5): x = x - f.gradient(x)
p.set_performance_metric(f(x) - f(xs))
tau = p.solve(verbose=0)

>>> print(f"{tau:.5f}")
0.04545
```

Example: gradient method on smooth convex functions

1.1 Gradient descent

PEPit.examples.unconstrained_convex_minimization.wc_gradient_descent(*L, gamma, n, wrapper='cvxpy', solver=None, verbose=1*) [source]

Consider the convex minimization problem

$$f_\star \triangleq \min_x f(x),$$

where f is L-smooth and convex.

This code computes a worst-case guarantee for **gradient descent** with fixed step-size γ. That is, it computes the smallest possible $\tau(n, L, \gamma)$ such that the guarantee

$$f(x_n) - f_\star \leqslant \tau(n, L, \gamma)\|x_0 - x_\star\|^2$$

is valid, where x_n is the output of gradient descent with fixed step-size γ, and where $x_\star$ is a minimizer of f.

In short, for given values of n, L, and γ, $\tau(n, L, \gamma)$ is computed as the worst-case value of $f(x_n) - f_\star$ when $\|x_0 - x_\star\|^2 \leqslant 1$.

Algorithm: Gradient descent is described by

$$x_{t+1} = x_t - \gamma \nabla f(x_t),$$

where γ is a step-size.

Theoretical guarantee: When $\gamma \leqslant \frac{1}{L}$, the **tight** theoretical guarantee can be found in [1, Theorem 3.1]:

$$f(x_n) - f_\star \leqslant \frac{L}{4nL\gamma + 2}\|x_0 - x_\star\|^2,$$

which is tight on some Huber loss functions.

[PEPit documentation]

Code: gradient method on smooth convex functions

```python
# Instantiate PEP
problem = PEP()

# Declare a smooth convex function
func = problem.declare_function(SmoothConvexFunction, L=L)

# Start by defining its unique optimal point xs = x_* and corresponding function value fs = f_*
xs = func.stationary_point()
fs = func(xs)

# Then define the starting point x0 of the algorithm
x0 = problem.set_initial_point()

# Set the initial constraint that is the distance between x0 and x^*
problem.set_initial_condition((x0 - xs) ** 2 <= 1)

# Run n steps of the GD method
x = x0
for _ in range(n):
    x = x - gamma * func.gradient(x)

# Set the performance metric to the function values accuracy
problem.set_performance_metric(func(x) - fs)

# Solve the PEP
pepit_verbose = max(verbose, 0)
pepit_tau = problem.solve(wrapper=wrapper, solver=solver, verbose=pepit_verbose)

# Compute theoretical guarantee (for comparison)
theoretical_tau = L / (2 * (2 * n * L * gamma + 1))

# Print conclusion if required
if verbose != -1:
    print('*** Example file: worst-case performance of gradient descent with fixed step-sizes ***')
    print('\tPEPit guarantee:\t f(x_n)-f_* <= {:.6} ||x_0 - x_*||^2'.format(pepit_tau))
    print('\tTheoretical guarantee:\t f(x_n)-f_* <= {:.6} ||x_0 - x_*||^2'.format(theoretical_tau))

# Return the worst-case guarantee of the evaluated method (and the reference theoretical value)
return pepit_tau, theoretical_tau
```

[PEPit documentation]

Output: gradient method on smooth convex functions

```
>>> L = 3
>>> pepit_tau, theoretical_tau = wc_gradient_descent(L=L, gamma=1 / L, n=4, wrapper="cvxpy", solver=None, verbose=1)
(PEPit) Setting up the problem: size of the Gram matrix: 7x7
(PEPit) Setting up the problem: performance measure is the minimum of 1 element(s)
(PEPit) Setting up the problem: Adding initial conditions and general constraints ...
(PEPit) Setting up the problem: initial conditions and general constraints (1 constraint(s) added)
(PEPit) Setting up the problem: interpolation conditions for 1 function(s)
                        Function 1 : Adding 30 scalar constraint(s) ...
                        Function 1 : 30 scalar constraint(s) added
(PEPit) Setting up the problem: additional constraints for 0 function(s)
(PEPit) Compiling SDP
(PEPit) Calling SDP solver
(PEPit) Solver status: optimal (wrapper:cvxpy, solver: MOSEK); optimal value: 0.1666666649793714
(PEPit) Primal feasibility check:
                The solver found a Gram matrix that is positive semi-definite up to an error of 6.325029629397552e-10
                All the primal scalar constraints are verified up to an error of 6.63361402267193e-09
(PEPit) Dual feasibility check:
                The solver found a residual matrix that is positive semi-definite
                All the dual scalar values associated with inequality constraints are nonnegative up to an error of
                7.0696174975682125e-09
(PEPit) The worst-case guarantee proof is perfectly reconstituted up to an error of 7.547097551421911e-08
(PEPit) Final upper bound (dual): 0.1666666733194192 and lower bound (primal example): 0.1666666649793714
(PEPit) Duality gap: absolute: 8.340047791266514e-09 and relative: 5.004028725419552e-08
*** Example file: worst-case performance of gradient descent with fixed step-sizes ***
        PEPit guarantee:         f(x_n)-f_* <= 0.166667 ||x_0 - x_*||^2
        Theoretical guarantee:   f(x_n)-f_* <= 0.166667 ||x_0 - x_*||^2
```

[PEPit documentation]

Many other frameworks

Tool / workflow	Main role
PESTO	Worst-case PEPs in MATLAB/YALMIP.
PEPit.jl	Worst-case PEPs in native Julia/JuMP.
PEPFlow	Interactive primal/dual and symbolic proof work.
AutoLyap / AutoLyap.jl	SDP search for Lyapunov proofs (Python/Julia).
BnB-PEP	Global optimization of method coefficients.
Peppy	LLM-assisted analytic proofs, checked by SymPy.
AutoOPT	Method design with BnB-PEP, LLMs and Lean.
OSPEP	Operator splitting and parameter selection.
PEPStructuredSets	PEPs for smooth and strongly convex sets.
GPEP (experimental)	Nonlinear models, heuristic search (uncertified).

Many other frameworks

Tool / workflow	Main role
PESTO	Worst-case PEPs in MATLAB/YALMIP.
PEPit.jl	Worst-case PEPs in native Julia/JuMP.
PEPFlow	Interactive primal/dual and symbolic proof work.
AutoLyap / AutoL	…lia).
BnB-PEP	…
Peppy	…ymPy.
AutoOPT	Method design with BnB-PEP, LLMs and Lean.
OSPEP	Operator splitting and parameter selection.
PEPStructuredSets	PEPs for smooth and strongly convex sets.
GPEP (experimental)	Nonlinear models, heuristic search (uncertified).

Claude demo!

Outline

Convergence bounds in optimization

What is Performance Estimation?

How does PEP work?

A selection of PEP results

Open questions

Conclusions References

Gradient method for smooth convex functions

Constant step h/L, $h \in [0, 2]$: exact final function-gap bound

$$\max f(x_N) - f^* = \frac{LR^2}{2} \max\left(\frac{1}{2Nh+1}, (1-h)^{2N}\right)$$

We actually know

- analytical expression for worst-case performance
- analytical expression of worst-case function (Huber/quadratic)
- analytical expression of dual multipliers

Even then a rigorous sum-of-squares proof can be hard to obtain!

- [Drori–Teboulle, 2014] for $h \leq 1$ (6 pages of tedious algebra)
- [Teboulle–Vaisbourd, 2023] for $h \leq 1.5$
- [Rotaru–G–Patrinos, 2026] [J. Kim, 2024] for $h \leq 2$

Dealing with smooth strongly convex functions

$0 < \mu < L$. The class $\mathcal{F}_{\mu,L}$ has lower curvature μ and upper curvature L.

Necessary and sufficient interpolation conditions

The data $\{(x_i, g_i, f_i)\}_{i \in S}$ are $\mathcal{F}_{\mu,L}$-interpolable if and only if

$$f_i - f_j - g_j^\top (x_i - x_j) \geq \frac{1}{2(1-\mu/L)} \left(\frac{1}{L} \|g_i - g_j\|^2 + \mu \|x_i - x_j\|^2 - \frac{2\mu}{L} (g_i - g_j)^\top (x_i - x_j) \right)$$

for all $i, j \in S$.

At $\mu = 0$, this becomes the smooth convex interpolation condition.

The same conditions also hold for $\mu < 0$.

[Taylor–Hendrickx–G, 2017a]

Gradient method for smooth strongly convex functions

$$x_{i+1} = x_i - \frac{h}{L}\nabla f(x_i), \qquad 0 < h < 2$$

Linear convergence for all performance criteria, with same rate
$\rho = \max\{(1-h)^2, (1-\frac{\mu}{L}h)^2\}$ [Taylor–Hendrickx–G, 2018]

$$\begin{aligned} \|x_k - x_*\|^2 &\leq \rho^k \|x_0 - x_*\|^2 \\ \|\nabla f(x_k)\|^2 &\leq \rho^k \|\nabla f(x_0)\|^2 \\ f(x_k) - f(x_*) &\leq \rho^k \big(f(x_0) - f(x_*)\big) \end{aligned}$$

- All results with fully rigorous PEP-type mathematical proofs
  (pure linear rates $\rightarrow$ sufficient to prove them for one step)

Gradient method for smooth strongly convex functions

$$x_{i+1} = x_i - \frac{h}{L}\nabla f(x_i), \qquad 0 < h < 2$$

Linear convergence for all performance criteria, with same rate $\rho = \max\{(1-h)^2, (1-\frac{\mu}{L}h)^2\}$ [Taylor–Hendrickx–G, 2018]

$$\begin{aligned} \|x_k - x_*\|^2 &\leq \rho^k \|x_0 - x_*\|^2 \\ \|\nabla f(x_k)\|^2 &\leq \rho^k \|\nabla f(x_0)\|^2 \\ f(x_k) - f(x_*) &\leq \rho^k \big(f(x_0) - f(x_*)\big) \end{aligned}$$

- All results with fully rigorous PEP-type mathematical proofs (pure linear rates $\rightarrow$ sufficient to prove them for one step)
- Optimal normalized stepsize is $h^* = \frac{2L}{L+\mu}$ with $\rho^* = \left(\frac{L-\mu}{L+\mu}\right)^2$

Gradient method for smooth strongly convex functions

$$x_{i+1} = x_i - \frac{h}{L}\nabla f(x_i), \qquad 0 < h < 2$$

Linear convergence for all performance criteria, with same rate $\rho = \max\{(1-h)^2, (1-\frac{\mu}{L}h)^2\}$ [Taylor–Hendrickx–G, 2018]

$$\begin{aligned} \|x_k - x_*\|^2 &\leq \rho^k \|x_0 - x_*\|^2 \\ \|\nabla f(x_k)\|^2 &\leq \rho^k \|\nabla f(x_0)\|^2 \\ f(x_k) - f(x_*) &\leq \rho^k \big(f(x_0) - f(x_*)\big) \end{aligned}$$

- All results with fully rigorous PEP-type mathematical proofs (pure linear rates $\rightarrow$ sufficient to prove them for one step)
- Optimal normalized stepsize is $h^* = \frac{2L}{L+\mu}$ with $\rho^* = \left(\frac{L-\mu}{L+\mu}\right)^2$
- Worst-case functions are 1D quadratics $\frac{\mu}{2}x^2$ and $\frac{L}{2}x^2$

Gradient method for smooth strongly convex functions

$$x_{i+1} = x_i - \frac{h}{L}\nabla f(x_i), \qquad 0 < h < 2$$

Linear convergence for all performance criteria, with same rate
$\rho = \max\{(1-h)^2, (1-\frac{\mu}{L}h)^2\}$ [Taylor–Hendrickx–G, 2018]

$$\begin{aligned} \|x_k - x_*\|^2 &\leq \rho^k \|x_0 - x_*\|^2 \\ \|\nabla f(x_k)\|^2 &\leq \rho^k \|\nabla f(x_0)\|^2 \\ f(x_k) - f(x_*) &\leq \rho^k \big(f(x_0) - f(x_*)\big) \end{aligned}$$

- All results with fully rigorous PEP-type mathematical proofs
  (pure linear rates $\rightarrow$ sufficient to prove them for one step)
- Optimal normalized stepsize is $h^* = \frac{2L}{L+\mu}$ with $\rho^* = \left(\frac{L-\mu}{L+\mu}\right)^2$
- Worst-case functions are 1D quadratics $\frac{\mu}{2}x^2$ and $\frac{L}{2}x^2$
- Projected/proximal gradient: same rates for the composite gap, distance and optimality residual

Exact line-search does not improve the worst case

Gradient descent with exact line search

Let f be L-smooth and μ-strongly convex; write $g_k = \nabla f(x_k)$.

$$x_{k+1} = x_k - \gamma_k g_k, \qquad \gamma_k \in \arg\min_{\gamma \geq 0} f(x_k - \gamma g_k).$$

Worst-case convergence

Same worst-case rate as the best constant step $\gamma^* = \frac{2}{L+\mu}$

$$f(x_N) - f(x_*) \leq \left(\frac{L-\mu}{L+\mu}\right)^{2N} \big(f(x_0) - f(x_*)\big).$$

Trick to handle line-search: relax using optimality, $\langle g_{k+1}, g_k \rangle = 0$.
Exact PEP relaxation; attained by a 2D quadratic. [de Klerk–G–Taylor, 2017]

Dealing with smooth nonconvex functions

Smooth nonconvex actually quite similar to smooth convex:

$$\text{smooth nonconvex functions} \Leftrightarrow \text{curvature must belong to } [-L, L]$$

To tackle smooth nonconvex functions (class $\mathcal{F}_{-L,L}$) we only need one new ingredient: suitable interpolation conditions

It turns out that interpolation conditions for smooth strongly convex functions in class $\mathcal{F}_{\mu,L}$ also work, with exactly the same expressions, when μ is negative !

Hence smooth nonconvex interpolations conditions are obtained simply by taking $\mu = -L$ [Taylor–Hendrickx–G, 2017b]

This is used in [Abbaszadehpeivasti et al., 2022] to derive tight rates for gradient method on smooth nonconvex objectives with stepsize $\gamma \leq \frac{\sqrt{3}}{L}$

Gradient method for smooth functions

The following rates, with $\kappa = \mu/L$, are all

- exact, with explicit analytical expressions,
- established using mathematical proofs (PEP-inspired),
- cover all constant stepsize schedules in the standard stepsize range $[0, \frac{2}{L}]$,
- cover convex, strongly convex, nonconvex and hypoconvex objective functions in a unified way
- now measure gradient norm from an initial function gap: $\|\nabla f\|^2$
  (for last iterate, or smallest among all iterates)

[Rotaru–G–Patrinos, 2026]

Exact rates for gradient method in (strongly) convex case

Constant step γ, $N \geq 1$, $t = \gamma L \in [0, 2]$, and $\kappa = \mu/L$.

Theorem (Convex: $f \in \mathcal{F}_{0,L}$)

$$\frac{\|\nabla f(x_N)\|^2}{2L} \leq \big(f(x_0) - f(x_*)\big) \max\left\{\frac{1}{1+2Nt}, (1-t)^{2N}\right\}.$$

Theorem (Strongly convex: $f \in \mathcal{F}_{\mu,L}$, $0 < \kappa \leq 1$)

$$\frac{\|\nabla f(x_N)\|^2}{2L} \leq \big(f(x_0) - f(x_*)\big) \max\left\{\frac{\kappa(1-\kappa t)^{2N}}{1-(1-\kappa)(1-\kappa t)^{2N}}, (1-t)^{2N}\right\}.$$

As $\kappa \to 0$, the first term tends to $(1+2Nt)^{-1}$.

Worst of two simple functions (quadratic/Huber), but the proof is *not* simple.

[Rotaru–G–Patrinos, 2026]

Exact rates for gradient method in nonconvex case

Constant step γ, $N \geq 1$, $t = \gamma L \in [0, 2]$, and $\kappa = \mu/L < 0$.

Theorem (Weakly convex (hypoconvex): $f \in \mathcal{F}_{\mu,L}$)

$$\frac{1}{2L} \min_{0 \leq i \leq N} \|\nabla f(x_i)\|^2 \leq \big(f(x_0) - f(x_*)\big) \max\left\{ \frac{1}{1 + t\, P_N(t, \kappa t)},\ (1-t)^{2N} \right\},$$

where

$$P_N(t, \kappa t) = \begin{cases} \left(2 + \dfrac{\kappa t}{1-\kappa}\right) N, & 0 \leq t \leq 1, \\ \dfrac{(2-t)(2-\kappa t)}{2 - t - \kappa t} \left(N + \dfrac{\kappa t}{1-\kappa} \displaystyle\sum_{k=1}^{N} \big[T_k(t, \kappa)\big]_+ \right), & 1 < t \leq 2. \end{cases}$$

As $t \to 1$, $[T_k(t,\kappa)]_+ \to 0$; hence $P_N(1, \kappa) = N(2-\kappa)/(1-\kappa)$.

At $t = 0$ or $t = 2$, the rate factor is 1. [Rotaru–G–Patrinos, 2026]

Comments on exact rates for gradient method

- First proof of a tight convergence rate for the gradient method covering the full range of constant stepsizes (i.e. $\gamma \in [0, \frac{2}{L}]$)

- In (strongly) convex case: simple expression for rate
  (worst among two simple functions)
  but proof is unexpectedly difficult
  (reason: hidden distinct regimes behind a single rate)
- In nonconvex/hypoconvex case: very complicated rates
  (interpolate continuously between convex and nonconvex)

- To cover full range need N distinct proofs/regimes
- Key element for tightness: use of distance-two inequalities

[Rotaru–G–Patrinos, 2026]

Optimized first-order methods

Optimized Gradient Method (OGM)

Convex L-smooth f, $R = \|x_0 - x_*\|$, and $N \geq 1$ iterations.
Parameters: $\theta_0 = 1$, with

$$\theta_k = \frac{1 + \sqrt{1 + 4\theta_{k-1}^2}}{2} \quad (1 \leq k < N), \qquad \theta_N = \frac{1 + \sqrt{1 + 8\theta_{N-1}^2}}{2}.$$

Initialize $y_0 = x_0$. For $k = 0, \ldots, N-1$,

$$\begin{aligned} y_{k+1} &= x_k - \frac{1}{L}\nabla f(x_k), \\ x_{k+1} &= y_{k+1} + \frac{\theta_k - 1}{\theta_{k+1}}(y_{k+1} - y_k) + \frac{\theta_k}{\theta_{k+1}}(y_{k+1} - x_k). \end{aligned}$$

One gradient evaluation per iteration; only a few vectors to store.

[Kim–Fessler, 2016, OGM1]

Optimized Gradient Method (OGM)

Convex L-smooth f, $R = \|x_0 - x_*\|$, and $N \geq 1$ iterations.

Exact worst-case rate

$$f(x_N) - f_* \leq \frac{LR^2}{2\theta_N^2} \leq \frac{LR^2}{(N+1)^2}.$$

Exact minimax optimum. Same $1/N^2$ order as FGM; half the leading constant.

[Kim–Fessler, 2016] [Drori, 2017]

Strongly convex analogue: ITEM

The Information-Theoretic Exact Method: optimal distance-to-distance convergence.

[Taylor–Drori, 2023]

OGM-G: optimize the gradient norm

Convex L-smooth f, $\Delta_0 = f(x_0) - f_*$, and $N \geq 1$ iterations.
Reverse the OGM parameters: $t_k = \theta_{N-k}$, so $t_N = 1$.
Initialize $y_0 = x_0$. For $k = 0, \ldots, N-1$,

$$y_{k+1} = x_k - \frac{1}{L}\nabla f(x_k),$$
$$x_{k+1} = y_{k+1} + \beta_k(y_{k+1} - y_k) + \gamma_k(y_{k+1} - x_k),$$

where

$$\beta_k = \frac{(t_k - 1)(2t_{k+1} - 1)}{t_k(2t_k - 1)}, \qquad \gamma_k = \frac{2t_{k+1} - 1}{2t_k - 1}.$$

The horizon N is fixed in advance.

[Kim–Fessler, 2021]

OGM-G: optimize the gradient norm

Convex L-smooth f, $\Delta_0 = f(x_0) - f_*$, and $N \geq 1$ iterations.

Tight worst-case gradient bound

$$\frac{\|\nabla f(x_N)\|^2}{2L} \leq \frac{\Delta_0}{t_0^2} = \frac{\Delta_0}{\theta_N^2}.$$

Optimal $1/N^2$ order. First log-free bound from an initial function gap. [Kim–Fessler, 2021]

Recent optimized methods

Lemniscate acceleration: $\|\nabla f(x_N)\|^2 = O(L^2R^2/N^4)$.

ITEM-f: strongly convex function-gap-to-function-gap convergence.
[Kim et al., 2026]

Proximal, projected and coordinate methods

Projected gradient and proximal methods

We can actually handle with little extra effort

- first-order methods for smooth constrained convex optimization i.e. express projection steps in our formulation

$$x_{k+1} = P_C\left[x_k - \frac{1}{L}\nabla f(x_k)\right]$$

- proximal algorithms i.e. express proximal steps

$$x_{k+1} = \operatorname{prox}_{f/L}(x_k) = \arg\min_u \left(f(u) + \frac{L}{2}\|u - x_k\|^2\right)$$

- composite minimization: $\min f(x) + h(x)$ where f is smooth and h is proximable, using proximal gradient method

$$x_{k+1} = \operatorname{prox}_{h/L}(x_k - \tfrac{1}{L}\nabla f(x_k))$$

[Taylor–Hendrickx–G, 2017b]

Projected and proximal gradient: formulation and rate

A proximal step is a linear oracle relation

$$x_+ = \mathrm{prox}_{\gamma h}(y) \quad \Longleftrightarrow \quad x_+ + \gamma s_+ = y, \qquad s_+ \in \partial h(x_+).$$

Proximal gradient combines two steps:

$$y = x - \gamma \nabla f(x), \qquad x_+ = \mathrm{prox}_{\gamma h}(y).$$

Add interpolation conditions for f and h.

Projected gradient is the case $h = \mathbb{I}_C$. [Taylor–Hendrickx–G, 2017b]

Projected and proximal gradient: formulation and rate

$F = f + h$: f is L-smooth and μ-strongly convex, h is closed convex, and $0 < \mu \leq L$.

$$x_{k+1} = \mathrm{prox}_{\gamma h}\big(x_k - \gamma \nabla f(x_k)\big), \qquad \gamma = \frac{2}{L+\mu}.$$

Exact worst-case rate, $N \geq 1$

$$F(x_N) - F_* \leq \left(\frac{L-\mu}{L+\mu}\right)^{2N} \big(F(x_0) - F_*\big).$$

Includes projected gradient. Tight already for a quadratic with $h = 0$.

[Taylor–Hendrickx–G, 2018, Thm. 2.1]

Coordinate methods: useful bounds, not yet exact PEPs

For convex block coordinate smooth f, update block b_k cyclically:

$$x_{k+1} = x_k - \frac{1}{L_{b_k}} U_{b_k} \nabla_{b_k} f(x_k).$$

Here U_b embeds block b into the full space.

A two-block result after $K \geq 1$ cycles

$$\min_{1 \leq i \leq 2K-1} \sum_{b=1}^{2} \frac{\|\nabla_b f(x_i)\|^2}{L_b} \leq \frac{2\big(f(x_0) - f(x_{2K})\big)}{2K-1}.$$

PEP also gives improved numerical upper bounds for cyclic coordinate descent, alternating minimization, and cyclic accelerated coordinate descent.

Tightness is not guaranteed: the available finite interpolation conditions are necessary but not sufficient.

[Kamri et al., 2025, Thm. 3.3]

Nonsmooth optimization

Nonsmooth optimization: subgradient method

First-order methods can deal with nonsmooth convex functions using the concept of subgradient

$$g \in \partial f(x) \quad \Leftrightarrow \quad f(y) \geq f(x) + g^T(y - x) \text{ for all } y$$

Subgradient method starts from x_0 and performs for all $i \geq 0$

$$x_{i+1} = x_i - h_i g_i \text{ for some } g_i \in \partial f(x_i)$$

for some stepsize schedule $\{h_i\}_{i \geq 0}$

Worst-case rates on objective accuracy require

- Distance R from initial iterate x_0 to minimizer x^*
- Bound B on maximum norm of any subgradient $g \in \partial f(x)$

Results: average function value

Worst-case for fixed-step subgradient method

$$x_{i+1} = x_i - h(\tfrac{R}{B})g_i$$

applied to convex function with B-bounded subgradients

- For average value of iterates $\hat{f}_N = \frac{f(x_0)+f(x_1)+\ldots+f(x_N)}{N+1}$, tight worst-case is

$$\hat{f}_N - f(x_*) \leq \begin{cases} BR\big(\frac{1}{2}h + \frac{1}{2(N+1)}\frac{1}{h}\big) & \text{when } h \geq \frac{1}{N+1} \\ BR\big(1 - \frac{N}{2}h\big) & \text{when } h \leq \frac{1}{N+1} \end{cases}$$

  By convexity, $f(\bar{x}_N) \leq \hat{f}_N$.

- Optimal constant step-size is then $h^* = \frac{1}{\sqrt{N+1}}$ (belongs to "large step" case) leading to tight worst-case [Zamani–G, 2025]

$$\hat{f}_N - f(x_*) \leq \frac{BR}{\sqrt{N+1}}$$

Results: last iterate

- Define sequence $\{s_N\}_{N\geq 0} = \{1, 2, \frac{5}{2}, \frac{29}{10}, \ldots\}$ with
$$s_0 = 1, s_{i+1} = s_i + \frac{1}{s_i} \text{ for all } i \geq 0$$
- No closed form, s_N grows like $\sqrt{2(N+1) + \frac{1}{2}\log(N)}$, also appears in [Nesterov, 2009] for primal-dual subgradient
- For value of last iterate $f(x_N)$, tight worst-case is
$$f(x_N) - f(x_*) \leq \begin{cases} BR\left[(\frac{1}{2}s_N^2 - N)h + \frac{1}{2s_N^2}\frac{1}{h}\right] & \text{when } h \geq \frac{1}{s_N^2} \\ BR(1-Nh) & \text{when } h \leq \frac{1}{s_N^2} \end{cases}$$
- First sharp last-iterate rate for constant stepsizes [Zamani–G, 2025]
- [Harvey et al., 2019] prove a $\frac{\log N}{32\sqrt{N}}$ lower bound when $B = 1$ with stepsize $h_i = \frac{1}{\sqrt{i}}$, and prove a high probability $\mathcal{O}(\frac{\log N}{\sqrt{N}})$ upper bound in stochastic case

Results: optimal stepsize and variants

- To perform N subgradient iterations, optimal stepsize is then

$$h^* = \frac{1}{\sqrt{s_N^2(s_N^2 - 2N)}}$$

  and corresponding worst-case value satisfies

$$f(x_N) - f(x_*) \leq BR\sqrt{1 - \frac{2N}{s_N^2}} \lesssim BR \cdot \sqrt{\frac{1 + \frac{1}{4}\log(N)}{N+1}}$$

- Using suboptimal $h^\dagger = \frac{1}{\sqrt{N+1}}$ gives a slower asymptotic rate

$$f(x_N) - f(x_*) \leq BR \cdot \left(\frac{\frac{5}{4} + \frac{1}{4}\log(N)}{\sqrt{N+1}}\right)$$

- All results also hold for normalized update $x_{i+1} = x_i - hR\frac{g_i}{\|g_i\|}$

[Zamani–G, 2025]

Proof technique: PEP no longer needed!

A weighted-distance argument and Jensen's inequality give the key lemma.

Lemma ([Zamani–G, 2025]**)**

Suppose that $\hat{x} \in X$, $h_{N+1} > 0$ and weights v_k satisfy $0 < v_0 \leq v_1 \leq \cdots \leq v_N \leq v_{N+1}$ Then iterates of the subgradient methods with starting point $x^1 \in X$ generating $\{(x^k, g^k)\}$ satisfy

$$\sum_{k=1}^{N+1} \left(h_k v_k^2 - (v_k - v_{k-1}) \sum_{i=k}^{N+1} h_i v_i \right) \Big(f(x^k) - f(\hat{x}) \Big)$$
$$\leq \tfrac{v_0^2}{2} \underbrace{\left\| x^1 - \hat{x} \right\|^2}_{R} + \tfrac{1}{2} \sum_{k=1}^{N+1} h_k^2 v_k^2 \underbrace{\left\| g^k \right\|^2}_{B}$$

Still credit for the inspiration of the proof goes to PEP!

Optimal last-iterate subgradient method

- Using the following new linearly decreasing stepsize schedule

$$x_{k+1} = x_k - \frac{R}{B}\frac{N-k}{(N+1)^{3/2}}\, g_k, \qquad k = 0, \ldots, N-1$$

  leads to the optimal rate for the last iterate [Zamani–G, 2025]

$$f(x_N) - f(x_*) \leq \frac{BR}{\sqrt{N+1}}$$

  (improves $\frac{15BD}{\sqrt{N+1}}$ [Jain et al., 2021] for diameter D)

- Same proof technique, using key lemma with other weights v_k

- Schedule dependence on N is forced for optimal method
  (already impossible to find fixed stepsizes h_1 and h_2 that are optimal for both $N = 1$ and $N = 2$)

- Existence of a last-iterate optimal method with momentum?

Inexact gradients and linear mappings

Methods using inexact gradient

Instead of computing $x_{k+1} = x_k - \frac{1}{L}\nabla f(x_k)$ with exact gradient assume gradient is computed inexactly with bounded error

$$\tilde{g}_k \approx \nabla f(x_k) \text{ such that } \|\tilde{g}_k - \nabla f(x_k)\| \leq \Delta$$

Key technique: rewrite step with inexact gradient

$$x_{k+1} = x_k - \frac{1}{L}\tilde{g}_k$$

as

$$x_{k+1} = x_k - \frac{1}{L}\nabla f(x_k) - \frac{1}{L}\big(\tilde{g}_k - \nabla f(x_k)\big)$$

and then observe it is equivalent to

$$L(x_k - x_{k+1}) - \nabla f(x_k) = \tilde{g}_k - \nabla f(x_k)$$

which can be written using the assumption on the error as

$$\|L(x_k - x_{k+1}) - \nabla f(x_k)\| \leq \Delta$$

Leads to a convex SDP formulation (after squaring both sides) [de Klerk–G–Taylor, 2020]

Gradient method with relative inexactness

Instead of constant maximum error define gradient step with relative inexactness

$$x_{k+1} = x_k - \frac{h}{L} d_k$$

with

$$\|d_k - \nabla f(x_k)\| \leq \Delta \|\nabla f(x_k)\|$$

We can prove again linear convergence for all performance criteria,

$$\begin{aligned} \|x_k - x_*\|^2 &\leq \rho^k \|x_0 - x_*\|^2 \\ \|\nabla f(x_k)\|^2 &\leq \rho^k \|\nabla f(x_0)\|^2 \\ f(x_k) - f(x_*) &\leq \rho^k \big(f(x_0) - f(x_*)\big) \end{aligned}$$

with same rate depending on Δ [de Klerk–G–Taylor, 2020]

$$\rho = (1 - (1 - \Delta)\tfrac{\mu}{L} h)^2$$

(smooth strongly convex f, for h small enough and $0 \leq \Delta \leq \frac{2\mu}{L+\mu}$)

Methods involving linear mappings

[Bousselmi–Hendrickx–G, 2024]

We want to minimize $g(Mx)$ (alone/in composite objective)
where M is a linear mapping with some characteristics

- M symmetric and constraints on minimum/maximum eigenvalues $[\mu, L]$
- M non-symmetric (possibly rectangular) with maximum singular value S

A gradient step on $F(x) = g(Mx)$ requires gradient

$$\nabla F(x) = M^T \nabla g(Mx)$$

which can be decomposed as three successive operations:

$$\begin{aligned} y &= Mx \\ u &= \nabla g(y), \\ v &= M^T u = \nabla F(x). \end{aligned}$$

Gradient of function composed with linear mapping

In order to compute worst-case for methods involving

$$\begin{aligned} y &= Mx \\ u &= \nabla g(y), \\ v &= M^T u = \nabla F(x). \end{aligned}$$

we need to interpolate the following

$$\begin{aligned} y_i &= Mx_i, \\ u_i &= \nabla g(y_i), \\ v_i &= M^T u_i = \nabla F(x_i). \end{aligned}$$

New expressions: $y_i = Mx_i$ and $v_i = M^T u_i$
Requires interpolability of two sequences $\{x_i, y_i\}$ and $\{u_i, v_i\}$
by a linear mapping M and its transpose M^T
[Bousselmi–Hendrickx–G, 2024]

Interpolation theorem for linear mappings

Collect the sampled vectors as columns of X, Y, U, V.

Let $X \in \mathbb{R}^{m\times N_1}$, $Y \in \mathbb{R}^{n\times N_1}$, $U \in \mathbb{R}^{n\times N_2}$ and $V \in \mathbb{R}^{m\times N_2}$.
There exists $M \in \mathbb{R}^{n\times m}$ with $\|M\| \leq S$, $MX = Y$ and $M^T U = V$ if and only if

$$\begin{cases} X^T V = Y^T U, \\ Y^T Y \preceq S^2 X^T X, \\ V^T V \preceq S^2 U^T U. \end{cases}$$

(where $\preceq$ denotes the Löwner = positive semidefinite order)

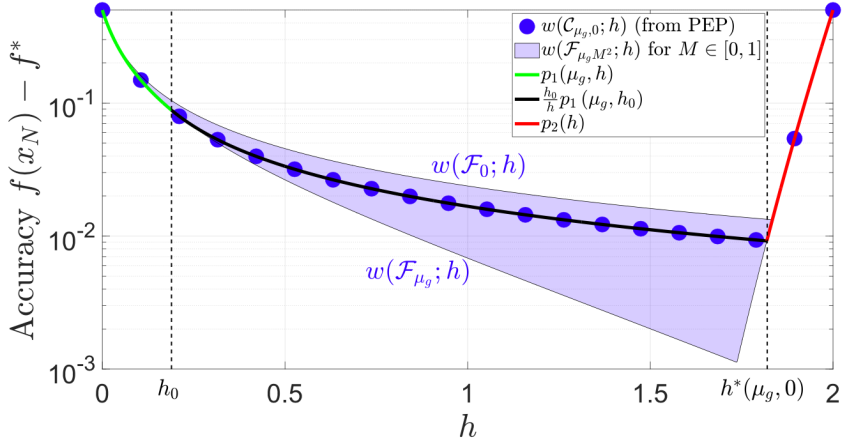
Moreover, if $U = X$ and $V = Y$ (resp. $V = -Y$), the interpolant matrix can be chosen symmetric (resp. skew-symmetric).

[Bousselmi–Hendrickx–G, 2024]

Gradient method: exploiting the linear map

$$\min g(Mx) \quad \text{vs.} \quad \min f(x)$$

g is strongly convex; $f = g \circ M$ need not be. [Bousselmi–Hendrickx–G, 2024]



Result: Chambolle–Pock method

Minimize $f(x) + g(Mx)$, with convex f, g and $\|M\| \leq L_M$.
Choose $\tau, \sigma > 0$ with $\tau\sigma L_M^2 \leq 1$.

$$x_{k+1} = \operatorname{prox}_{\tau f}(x_k - \tau M^T u_k),$$
$$u_{k+1} = \operatorname{prox}_{\sigma g^*}\big(u_k + \sigma M(2x_{k+1} - x_k)\big).$$

Here g^* is the convex conjugate of g. [Chambolle–Pock, 2011]

Average output after $N \geq 1$ updates

$$\bar{x}_N = \frac{1}{N}\sum_{k=1}^{N} x_k, \qquad \bar{u}_N = \frac{1}{N}\sum_{k=1}^{N} u_k.$$

Result: Chambolle–Pock method

Minimize $f(x) + g(Mx)$, with convex f, g and $\|M\| \leq L_M$.
The Lagrangian is $\mathcal{L}(x, u) = f(x) + u^T Mx - g^*(u)$.

Exact primal–dual gap bound

For any test pair (x, u),

$$\mathcal{L}(\bar{x}_N, u) - \mathcal{L}(x, \bar{u}_N) \leq \frac{\frac{1}{\tau}\|x - x_0\|^2 + \frac{1}{\sigma}\|u - u_0\|^2 - 2(u - u_0)^T M(x - x_0)}{2(N+1)}.$$

Tight, with an analytical proof. Improves the earlier denominator $2N$ to $2(N+1)$.

[Bousselmi–Hendrickx–G, 2024, Thm. 21]

Other algorithms analyzed with PEP

Gradient and momentum methods
Heavy-ball; Nesterov/FGM; OGM/OGM-G; ITEM; triple/robust momentum; exact line search; nonlinear conjugate gradient.

Proximal and constrained methods
Projected/proximal gradient; FISTA; (accelerated) proximal point; Frank–Wolfe; Bregman gradient and NoLips.

Randomized and block methods
Stochastic gradient; SAGA/Point-SAGA; random/cyclic coordinate descent; alternating minimization.

Splitting and primal–dual methods
ADMM; Douglas–Rachford; Peaceman–Rachford; Davis–Yin three-operator splitting; Chambolle–Pock/PDHG.

Fixed points and monotone operators
Krasnosel'skiĭ–Mann; Halpern; proximal point; extragradient/past extragradient; optimistic gradient.

Nonconvex and further extensions
DCA; weakly convex subgradient; inexact gradient/proximal methods; Newton and cubic regularization (univariate).

Results depend on the function/operator class, metric and variant; some use relaxations.

Outline

Convergence bounds in optimization

What is Performance Estimation?

How does PEP work?

A selection of PEP results

Open questions

Conclusions References

Reflections

- Automated procedure to compute worst-case rates
  But: is a PEP proof the final goal?

- More than once, these steps actually happened
  1. Numerical rate computed, analytical expression guessed/identified/confirmed
  2. Using insight provided by worst-case/proof, a closer look at rate/proof provides further intuition and new ideas
  3. Result can now be derived in standard way
     (and its “PEP” origin becomes unnoticeable!)

- Even when such simplifications are not found, shouldn’t the goal of mathematical optimization theory first and foremost be to increase our understanding?

Open questions: new function classes

Exact PEPs require necessary and sufficient interpolation conditions. Necessary conditions alone give valid upper bounds, which may be loose.

Example: convex block coordinate smooth functions

Each partial gradient $\nabla_b f$ is L_b-Lipschitz when only block b varies. The interpolation problem is open already for two blocks.

Known necessary conditions, for all data pairs (i, j) and blocks b:

$$f_i - f_j - \langle g_j, x_i - x_j \rangle \geq \frac{\|g_i^{(b)} - g_j^{(b)}\|^2}{2L_b}.$$

- These conditions are sufficient for **two data points**.
- They are **not sufficient for three or more points**: pairwise consistency does not ensure one global interpolant.

[Kamri et al., 2025, Thms. 2.3–2.4]

Open questions: Hessian-Lipschitz functions

For second-order methods, consider C^2 functions satisfying

$$\|\nabla^2 f(x) - \nabla^2 f(y)\|_{\mathrm{op}} \leq M\|x - y\|,$$

optionally with convexity: $\nabla^2 f(x) \succeq 0$.

What must be interpolated?

Finite second-order data $\{(x_i, f_i, g_i, H_i)\}$, with

$$f(x_i) = f_i, \qquad \nabla f(x_i) = g_i, \qquad \nabla^2 f(x_i) = H_i.$$

Known in dimension one: necessary and sufficient conditions, with and without convexity; tight analyses of Newton and cubic-regularized Newton variants.

[Rubbens et al., 2025]

Open: sharp, computationally usable interpolation conditions in several dimensions. Exact interpolation need not yield a convex SDP.

Open questions: can method design be convex?

Goal: a general exact convex formulation of $\min_H W_N(H)$, where $W_N(H)$ is the worst-case performance of method H.
Natural approach: optimize the dual PEP jointly over the method and its proof. Write $x_i = x_0 - L^{-1}\sum_{k<i} H_{ik} g_k$.
For smooth convex functions, interpolation gives

$$I_{ij} := f_i - f_j - \langle g_j, x_i - x_j\rangle - \frac{\|g_i - g_j\|^2}{2L} \geq 0.$$

Its dual weight $\lambda_{ij} \geq 0$ creates terms of the form

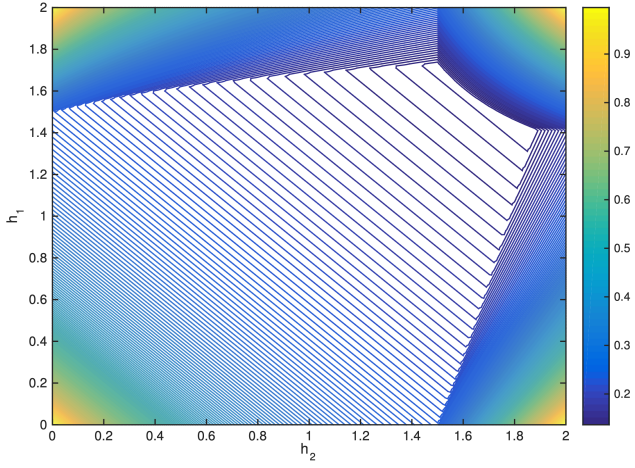
$$\lambda_{ij}\langle g_j, x_i - x_j\rangle = -\frac{1}{L}\sum_k \underbrace{\lambda_{ij}(H_{ik} - H_{jk})}_{\text{bilinear in proof and method}} \langle g_j, g_k\rangle,$$

with $H_{ik} = 0$ for $k \geq i$.

Fixed method: an SDP. **Joint design:** a nonconvex matrix inequality.
[Drori–Teboulle, 2014] [Kamri–Hendrickx–G, 2025]

Convergence of $N = 2$ steps of gradient method

[Daccache, 2019] Stepsizes (h_1, h_2), convex objective, $L = 1$-smooth, initial distance $\|x_0 - x_*\| \leq 1$; contour plot of $2(f(x_2) - f_*)$



Numerical design already produces useful methods

BnB-PEP

[Das Gupta et al., 2024]

Reformulate design as a nonconvex QCQP; combine local search with **branch-and-bound**.

Global bounds can certify optimality when the search closes the gap.

Main limitation: computational cost as the horizon grows.

Local design methods

[Kamri–Hendrickx–G, 2025]

Alternating minimization, first-order search, and **successive linearization** of the matrix inequality.

Memoryless and full methods; coordinate, inexact and cyclic gradient variants.

Efficient tuning, with no general global-optimality guarantee.

Each candidate can be analyzed by a fixed-method PEP. Proving optimal *design* also requires a matching lower bound across methods.

Convex design: important special cases

Distance-one inequalities

Using only adjacent-iterate inequalities
In some setttings gives a convex SDP and recursive recovery of the method
May give non tight bounds (i.e. already for the gradient method)
But allowed intial derivation of optimized methods!
[Drori–Teboulle, 2014] [Taylor–Drori, 2023]

Chronological inequalities / causal multipliers

Use only on half of interpolation inequalities, in the same direction (lower triangular multiplier matrix) leads to a convex SDP design problem
[Drori–Teboulle, 2014] [Abdelsamed et al., 2026]

Open: remove certificate restrictions while retaining convexity

Open question: why such low-dimensional worst cases?

Many PEPs admit worst cases that are one-dimensional or use only a few directions: quadratics, Huber-type functions, and piecewise quadratic examples.

Gram-matrix viewpoint

If $G = P^\top P$, the vectors in P can be realized in dimension

$$d = \operatorname{rank} G.$$

A low-rank optimal Gram matrix gives a low-dimensional worst-case instance.

However subgradient worst-case constructions use dimension $N + 1$, growing with the number of iterations. [Zamani–G, 2025]

- Which classes, methods and criteria force a uniformly small rank?
- Can the needed dimension be predicted before solving the PEP?
- Can low-dimensional examples explain the structure of optimal proofs?

Open question: adaptive algorithms

To keep a PEP semidefinite, algorithm constraints must be linear (allowing substitution) or admit an SDP representation. Fixed coefficients give linear iterate updates.

Barzilai–Borwein stepsizes

$$\gamma_k = \frac{\|s_k\|^2}{\langle s_k, y_k \rangle}, \qquad \begin{aligned} s_k &= x_k - x_{k-1}, \\ y_k &= g_k - g_{k-1}. \end{aligned}$$

Substitution in $x_{k+1} = x_k - \gamma_k g_k$ introduces products of unknown Gram entries.

[Barzilai–Borwein, 1988]

Methods with "if" decisions

Backtracking: if the decrease test fails, shrink the step and retry.

Adaptive restart: if the objective increases, reset momentum.

Branching gives a union of feasible sets, generally nonconvex.

Open: tractable PEPs that can encode adaptive rules.

Thank you!

PEP turns convergence analysis
into optimization

PEPit

`https://pepit.readthedocs.io/`

Tutorial and notebooks

`https://github.com/PerformanceEstimation/Tutorial-SIAM-OP26`

Slides and Claude demos will be available on
`https://perso.uclouvain.be/francois.glineur/`

Happy to discuss this with all of you!

Sessions organized at the 2027 EUROPT and ISMP conferences

References

- Foundations and classical results
- Gradient, optimized and subgradient methods
- Operators, function classes and method design
- Software and computational frameworks
- Further algorithms and operator methods

References: Foundations and classical results

[Nesterov, 2018] Yurii Nesterov (2018).
Lectures on Convex Optimization.
2nd ed., Springer Optimization and Its Applications, vol. 137. Springer.

[Nesterov, 1983] Yurii E. Nesterov (1983).
A method of solving a convex programming problem with convergence rate $O(1/k^2)$.
Doklady Akademii Nauk SSSR 269(3), 543–547; English translation: Soviet Mathematics Doklady 27(2), 372–376.

[Nemirovsky–Yudin, 1983] A. S. Nemirovsky, D. B. Yudin (1983).
Problem Complexity and Method Efficiency in Optimization.
Wiley-Interscience Series in Discrete Mathematics. John Wiley & Sons; translated by E. R. Dawson.

[Reference index]

References: Foundations and classical results

[Ang, 2023] Andersen Ang (2023).
Nesterov's Accelerated Gradient Descent on L-smooth convex function: Proof approach 1.
Lecture slides, version 4 November 2023; first draft 2 August 2017.

[Drori–Teboulle, 2014] Yoel Drori, Marc Teboulle (2014).
Performance of first-order methods for smooth convex minimization: a novel approach.
Mathematical Programming 145, 451–482.

[Taylor–Hendrickx–G, 2017a] Adrien B. Taylor, Julien M. Hendrickx, François Glineur (2017a).
Smooth strongly convex interpolation and exact worst-case performance of first-order methods.
Mathematical Programming 161(1–2), 307–345.

[Reference index]

References: Foundations and classical results

[Taylor–Hendrickx–G, 2017b] Adrien B. Taylor, Julien M. Hendrickx, François Glineur (2017b).
Exact Worst-Case Performance of First-Order Methods for Composite Convex Optimization.
SIAM Journal on Optimization 27(3), 1283–1313.

[Lessard et al., 2016] Laurent Lessard, Benjamin Recht, Andrew Packard (2016).
Analysis and Design of Optimization Algorithms via Integral Quadratic Constraints.
SIAM Journal on Optimization 26(1), 57–95.

[Polyak, 1960s] Boris T. Polyak (1967).
A general method for solving extremal problems.
Doklady Akademii Nauk SSSR 174(1), 33–36.

[Reference index]

References: Foundations and classical results

[Taylor–Bach, 2019] Adrien Taylor, Francis Bach (2019).
Stochastic first-order methods: non-asymptotic and computer-aided analyses via potential functions.
Proceedings of the 32nd Conference on Learning Theory, PMLR 99, 2934–2992.

[Drori–Teboulle, 2016] Yoel Drori, Marc Teboulle (2016).
An optimal variant of Kelley's cutting-plane method.
Mathematical Programming 160(1–2), 321–351.

[Barzilai–Borwein, 1988] Jonathan Barzilai, Jonathan M. Borwein (1988).
Two-Point Step Size Gradient Methods.
IMA Journal of Numerical Analysis 8(1), 141–148.

[Reference index]

References: Gradient and subgradient methods

[Teboulle–Vaisbourd, 2023] Marc Teboulle, Yakov Vaisbourd (2023).
An elementary approach to tight worst case complexity analysis of gradient based methods.
Mathematical Programming 201, 63–96.

[Rotaru–G–Patrinos, 2026] Teodor Rotaru, François Glineur, Panagiotis Patrinos (2026).
Exact worst-case convergence rates of gradient descent: a complete analysis for all constant stepsizes over nonconvex and convex functions.
Mathematical Programming, published online 26 May 2026.

[J. Kim, 2024] Jungbin Kim (2024).
A Proof of the Exact Convergence Rate of Gradient Descent.
arXiv:2412.04427; revised March 2025.

[Reference index]

References: Gradient and subgradient methods

[Taylor–Hendrickx–G, 2018] Adrien B. Taylor, Julien M. Hendrickx, François Glineur (2018).
Exact Worst-Case Convergence Rates of the Proximal Gradient Method for Composite Convex Minimization.
Journal of Optimization Theory and Applications 178, 455–476.

[de Klerk–G–Taylor, 2017] Etienne de Klerk, François Glineur, Adrien B. Taylor (2017).
On the worst-case complexity of the gradient method with exact line search for smooth strongly convex functions.
Optimization Letters 11, 1185–1199.

[Abbaszadehpeivasti et al., 2022] Hadi Abbaszadehpeivasti, Etienne de Klerk, Moslem Zamani (2022).
The exact worst-case convergence rate of the gradient method with fixed step lengths for L-smooth functions.
Optimization Letters 16, 1649–1661.

[Reference index]

References: Gradient and subgradient methods

[Kim–Fessler, 2016] Donghwan Kim, Jeffrey A. Fessler (2016).
Optimized first-order methods for smooth convex minimization.
Mathematical Programming 159(1), 81–107.

[Drori, 2017] Yoel Drori (2017).
The exact information-based complexity of smooth convex minimization.
Journal of Complexity 39, 1–16.

[Taylor–Drori, 2023] Adrien B. Taylor, Yoel Drori (2023).
An optimal gradient method for smooth strongly convex minimization.
Mathematical Programming 199, 557–594.

[Reference index]

References: Gradient and subgradient methods

[Kim–Fessler, 2021] Donghwan Kim, Jeffrey A. Fessler (2021).
Optimizing the Efficiency of First-Order Methods for Decreasing the Gradient of Smooth Convex Functions.
Journal of Optimization Theory and Applications 188(1), 192–219.

[Kim et al., 2026] Heechang Kim, Ernest K. Ryu, Shuvomoy Das Gupta (2026).
A Domain-Specific Harness for End-to-End Automation of Optimization Research.
arXiv:2608.07407.

[Nesterov, 2009] Yurii Nesterov (2009).
Primal-dual subgradient methods for convex problems.
Mathematical Programming 120(1), 221–259.

[Reference index]

References: Gradient and subgradient methods

[Harvey et al., 2019] Nicholas J. A. Harvey, Christopher Liaw, Yaniv Plan, Sikander Randhawa (2019).
Tight analyses for non-smooth stochastic gradient descent.
Proceedings of COLT, PMLR99, 1579–1613.

[Zamani–G, 2025] Moslem Zamani, François Glineur (2025).
Exact Convergence Rate of the Last Iterate in Subgradient Methods.
SIAM Journal on Optimization 35(3), 2182–2201.

[Jain et al., 2021] Prateek Jain, Dheeraj M. Nagaraj, Praneeth Netrapalli (2021).
Making the Last Iterate of SGD Information Theoretically Optimal.
SIAM Journal on Optimization 31(2), 1108–1130.

[Reference index]

References: Operators and method design

[de Klerk–G–Taylor, 2020] Etienne de Klerk, François Glineur, Adrien B. Taylor (2020).
Worst-Case Convergence Analysis of Inexact Gradient and Newton Methods Through Semidefinite Programming Performance Estimation.
SIAM Journal on Optimization 30(3), 2053–2082.

[Bousselmi–Hendrickx–G, 2024] Nizar Bousselmi, Julien M. Hendrickx, François Glineur (2024).
Interpolation Conditions for Linear Operators and Applications to Performance Estimation Problems.
SIAM Journal on Optimization 34(3), 3033–3063.

[Chambolle–Pock, 2011] Antonin Chambolle, Thomas Pock (2011).
A First-Order Primal-Dual Algorithm for Convex Problems with Applications to Imaging.
Journal of Mathematical Imaging and Vision 40, 120–145.

[Reference index]

References: Operators and method design

[Rubbens et al., 2025] Anne Rubbens, Nizar Bousselmi, Julien M. Hendrickx, François Glineur (2025).
Performance Estimation of second-order optimization methods on classes of univariate functions.
arXiv:2506.22764.

[Kamri et al., 2025] Yassine Kamri, François Glineur, Julien M. Hendrickx, Ion Necoara (2025).
On the Worst-Case Analysis of Cyclic Block Coordinate Descent type Algorithms.
arXiv:2507.16675, revised 2026.

[Das Gupta et al., 2024] Shuvomoy Das Gupta, Bart P. G. Van Parys, Ernest K. Ryu (2024).
Branch-and-bound performance estimation programming: A unified methodology for constructing optimal optimization methods.
Mathematical Programming 204, 567–639.

[Reference index]

References: Operators and method design

[Kamri–Hendrickx–G, 2025] Yassine Kamri, Julien M. Hendrickx, François Glineur (2025).
Numerical Design of Optimized First-Order Algorithms.
arXiv:2507.20773, revised 2026.

[Daccache, 2019] Antoine Daccache (2019).
Performance estimation of the gradient method with fixed arbitrary step sizes.
Master's thesis, UCLouvain.

[Abdelsamed et al., 2026] Adel Abdelsamed, Dennis Gramlich, Christian Ebenbauer, Carsten W. Scherer (2026).
Synthesis of the Transient Behaviour of Optimization Algorithms for Strongly Convex Objective Functions.
arXiv:2610.23206.

[Reference index]

References: Operators and method design

[Park–Ryu, 2024] Chanwoo Park, Ernest K. Ryu (2024).
Optimal First-Order Algorithms as a Function of Inequalities.
Journal of Machine Learning Research 25(51), 1–66.

[Das Gupta et al., 2025] Shuvomoy Das Gupta, Robert M. Freund, Xu Andy Sun, Adrien Taylor (2025).
Nonlinear conjugate gradient methods: worst-case convergence rates via computer-assisted analyses.
Mathematical Programming 213, 1–49.

[Reference index]

References: Software and frameworks

[Goujaud, Moucer, G, Hendrickx, Taylor, Dieuleveut, 2024] Baptiste Goujaud, Céline Moucer, François Glineur, Julien M. Hendrickx, Adrien B. Taylor, Aymeric Dieuleveut (2024).
PEPit: computer-assisted worst-case analyses of first-order optimization methods in Python.
Mathematical Programming Computation 16, 337–367.

[Taylor–Hendrickx–G, 2017c] Adrien B. Taylor, Julien M. Hendrickx, François Glineur (2017c).
Performance estimation toolbox (PESTO): Automated worst-case analysis of first-order optimization methods.
56th IEEE Conference on Decision and Control, 1278–1283.

[Das Gupta et al., PEPit.jl] Shuvomoy Das Gupta and contributors.
PEPit.jl: Native Julia implementation of the Performance Estimation Programming methodology.
Software repository; accessed 12 November 2026.

[Reference index]

References: Software and frameworks

[Suh et al., 2025] Jaewook J. Suh, Bicheng Ying, Xin Jiang, Edward Duc Hien Nguyen (2025).
PEPFlow: A Python Library for the Workflow of Performance Estimation of Optimization Algorithms.
NeurIPS Workshop on GPU-Accelerated and Scalable Optimization (ScaleOPT).

[Upadhyaya et al., 2025] Manu Upadhyaya, Shuvomoy Das Gupta, Adrien B. Taylor, Sebastian Banert, Pontus Giselsson (2025).
The AutoLyap software suite for computer-assisted Lyapunov analyses of first-order methods.
arXiv:2506.24076, revised March 2026.

[Suh et al., 2026] Jaewook J. Suh, TaeHo Yoon, Edward Duc Hien Nguyen, Bicheng Ying, Shiqian Ma (2026).
Peppy: An AI-Assisted Workflow for Tight Convergence Analysis of Optimization Algorithms.
arXiv:2609.35762.

[Reference index]

References: Software and frameworks

[Ryu et al., 2020] Ernest K. Ryu, Adrien B. Taylor, Carolina Bergeling, Pontus Giselsson (2020).
Operator Splitting Performance Estimation: Tight Contraction Factors and Optimal Parameter Selection.
SIAM Journal on Optimization 30(3), 2251–2271.

[Luner–Grimmer, 2024] Alan Luner, Benjamin Grimmer (2024).
Performance Estimation for Smooth and Strongly Convex Sets.
arXiv:2410.14811, revised June 2026.

[Reference index]

References: Software and frameworks

[Serré, GPEP] Gaëtan Serré.
GPEP: An experimental Python framework for computer-assisted worst-case analyses of first-order optimization methods.
Software repository; accessed 12 November 2026.

[PEPit documentation] PEPit contributors.
PEPit documentation: Examples.
Online example catalogue; accessed 12 November 2026.

[Reference index]

References: Further algorithms

[Zamani et al., 2024] Moslem Zamani, Hadi Abbaszadehpeivasti, Etienne de Klerk (2024).
The exact worst-case convergence rate of the alternating direction method of multipliers.
Mathematical Programming 208, 243–276.

[Abbaszadehpeivasti et al., 2024] Hadi Abbaszadehpeivasti, Etienne de Klerk, Moslem Zamani (2024).
On the Rate of Convergence of the Difference-of-Convex Algorithm (DCA).
Journal of Optimization Theory and Applications 202, 475–496.

[Lieder, 2021] Felix Lieder (2021).
On the convergence rate of the Halpern-iteration.
Optimization Letters 15, 405–418.

[Reference index]

References: Further algorithms

[Dragomir et al., 2022] Radu-Alexandru Dragomir, Adrien B. Taylor, Alexandre d'Aspremont, Jérôme Bolte (2022).
Optimal complexity and certification of Bregman first-order methods.
Mathematical Programming 194, 41–83.

[Gorbunov et al., 2022a] Eduard Gorbunov, Nicolas Loizou, Gauthier Gidel (2022a).
Extragradient Method: $O(1/K)$ Last-Iterate Convergence for Monotone Variational Inequalities and Connections With Cocoercivity.
AISTATS, Proceedings of Machine Learning Research 151, 366–402.

[Reference index]

References: Further algorithms

[Gorbunov et al., 2022b] Eduard Gorbunov, Adrien Taylor, Gauthier Gidel (2022b).
Last-Iterate Convergence of Optimistic Gradient Method for Monotone Variational Inequalities.
arXiv:2205.08446.

[D. Kim, 2021] Donghwan Kim (2021).
Accelerated proximal point method for maximally monotone operators.
Mathematical Programming 190, 57–87.

[Reference index]