Performance Estimation Problems (PEPs)

Systematic, principled, and computer-aided approaches
to the analysis and design of first-order optimization algorithms

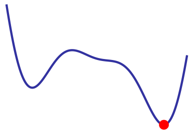
Aymeric Dieuleveut, Adrien Taylor

SMAI-MODE tutorial – March 2026

Context: numerical (continuous) optimization

Minimize $f : \mathbb{R}^d \to \mathbb{R}$ (e.g., with f continuous)

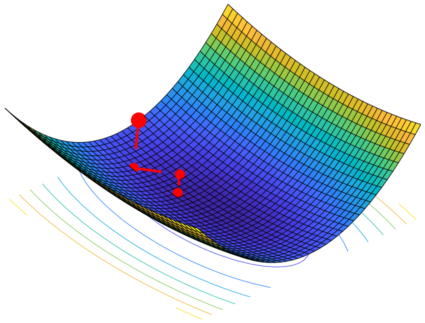
$$f(x_\star) \triangleq \min_{x \in \mathbb{R}^d} f(x).$$



Ubiquitous in applied mathematics and computer science.

Numerous applications for modeling (physics, economics), estimation (statistics, machine learning), decisions (control, operations research).

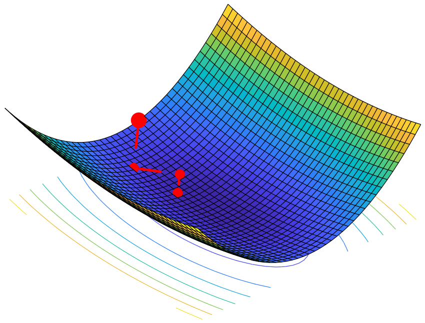
Usually solved via **iterative algorithm** generating sequence $x_0, x_1, \dots, x_N$.



Gradient descent (stepsize α)

```
for $k = 0, 1, \dots$ do
    $x_{k+1} = x_k - \alpha \nabla f(x_k)$
end for
```

Usually solved via **iterative algorithm** generating sequence $x_0, x_1, \dots, x_N$.

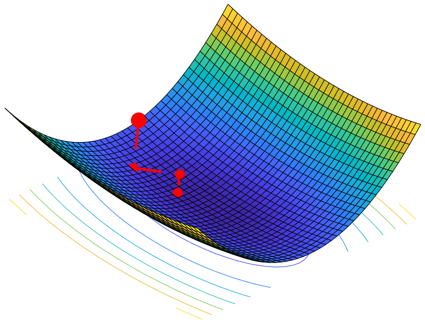


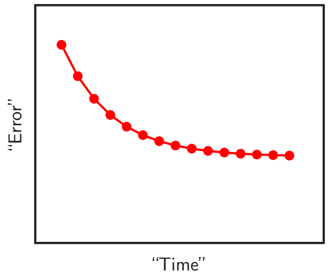
```
**Gradient descent** (stepsize $\alpha$)
    **for** $k = 0, 1, \dots$ **do**
        $x_{k+1} = x_k - \alpha \nabla f(x_k)$
    **end for**
```

What to expect from the output of the algorithm?

For instance: **bounds** on certain notions of "error": $f(x_k) - f(x_\star)$, $\|x_k - x_\star\|$, $\|\nabla f(x_k)\|$, etc.

Usually solved via **iterative algorithm** generating sequence $x_0, x_1, \ldots, x_N$.

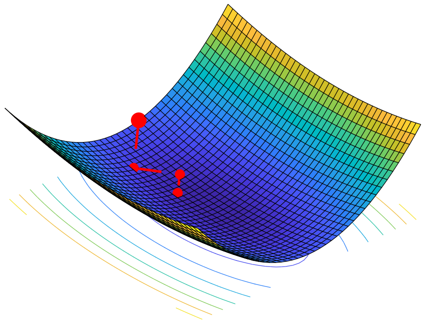


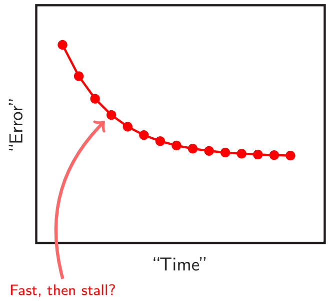


What to expect from the output of the algorithm?

For instance: **bounds** on certain notions of "error": $f(x_k) - f(x_\star)$, $\|x_k - x_\star\|$, $\|\nabla f(x_k)\|$, etc.

Usually solved via **iterative algorithm** generating sequence $x_0, x_1, \ldots, x_N$.

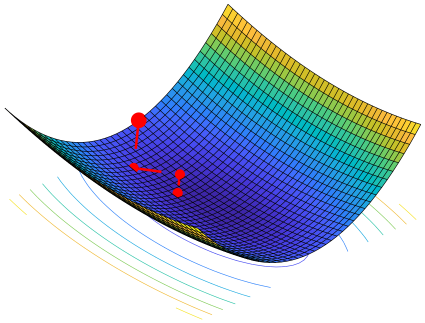


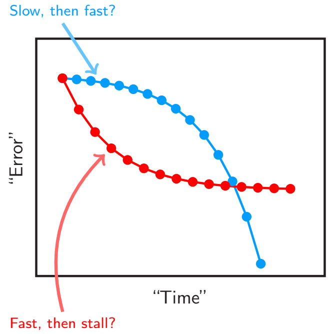


What to expect from the output of the algorithm?

For instance: **bounds** on certain notions of "error": $f(x_k) - f(x_\star)$, $\|x_k - x_\star\|$, $\|\nabla f(x_k)\|$, etc.

Usually solved via **iterative algorithm** generating sequence $x_0, x_1, \ldots, x_N$.

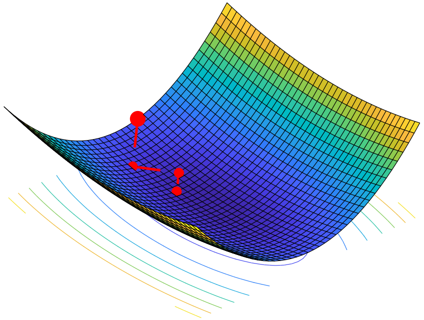


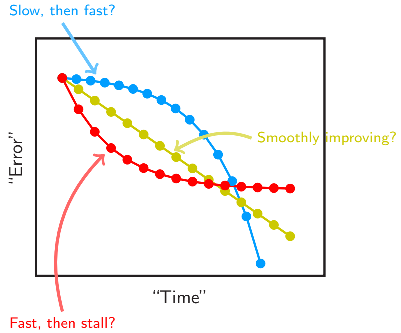


What to expect from the output of the algorithm?

For instance: **bounds** on certain notions of "error": $f(x_k) - f(x_\star)$, $\|x_k - x_\star\|$, $\|\nabla f(x_k)\|$, etc.

Usually solved via **iterative algorithm** generating sequence $x_0, x_1, \ldots, x_N$.

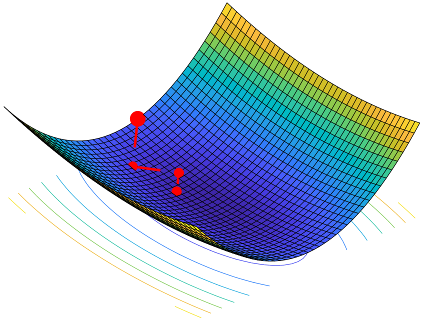


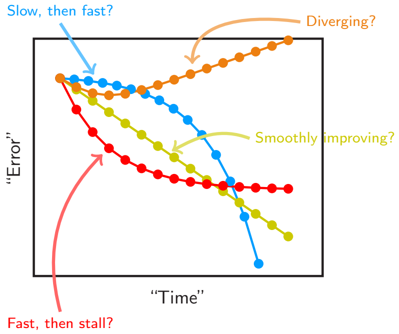


What to expect from the output of the algorithm?

For instance: **bounds** on certain notions of "error": $f(x_k) - f(x_\star)$, $\|x_k - x_\star\|$, $\|\nabla f(x_k)\|$, etc.

Usually solved via **iterative algorithm** generating sequence $x_0, x_1, \ldots, x_N$.





What to expect from the output of the algorithm?

For instance: **bounds** on certain notions of "error": $f(x_k) - f(x_\star)$, $\|x_k - x_\star\|$, $\|\nabla f(x_k)\|$, etc.

How to show that an algorithm works?

How to show that an algorithm works?

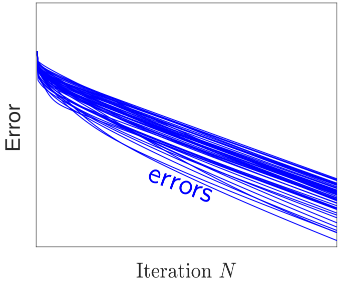
- ⋄ Assumptions (no free lunch).

How to show that an algorithm works?

- ◇ Assumptions (no free lunch).
- ◇ Here: worst-case perspective.

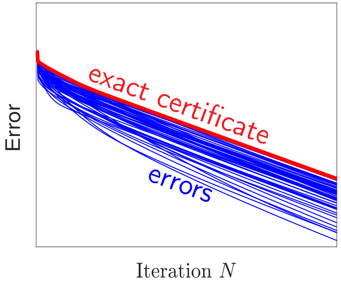
How to show that an algorithm works?

- ◇ Assumptions (no free lunch).
- ◇ Here: worst-case perspective.



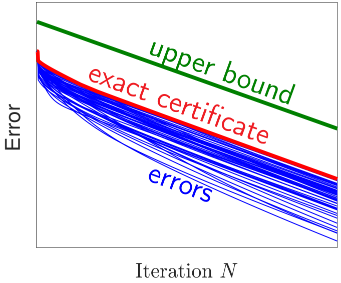
How to show that an algorithm works?

- ◇ Assumptions (no free lunch).
- ◇ Here: worst-case perspective.



How to show that an algorithm works?

- ◇ Assumptions (no free lunch).
- ◇ Here: worst-case perspective.



Organization and learning outcomes (1/3)

Observations:

- ◇ Complexity analyses / convergence proofs for first-order optimization methods often follow very similar (obscure?) patterns.
- ◇ ... these proofs are rarely intuitive—rely on combining (many) inequalities.
- ◇ Many variations (algorithms, function, key inequalities, etc.).

Performance estimation framework:

- ◇ helps understand proof structures of worst-case analyses,
- ◇ provides principled way to explore complexity bounds/derive such analyses,
- ◇ mathematical proofs via numerical experiments and symbolic computations.

Organization and learning outcomes (2/3)

This tutorial:

- Interactive introduction to *performance estimation problems* (PEPs).
- Learn how to use numerical PEPs to assess performance of simple first-order algorithms.
- Learn how to exploit PEP outputs to construct Lyapunov-based convergence proofs.
- Toy symbolic computations for search & verifying obtained guarantees.
- Basics of leveraging PEPs for algorithm design.

Organization and learning outcomes (3/3)

0. Organization
1. Base performance estimation problems
   - Basic examples
   - first-order oracles, duality, interpolation, semidefinite programming.
2. Structured convergence & non-convergence certificates
   - Using PEPs to search for structured proofs (convergence).
   - Using PEPs to search for cycles (no convergence).
3. Numerical design of optimized algorithms
   - Using PEPs to design algorithms.

For all parts: examples, guided hands-on notebooks, and Q&A.

Ressources

`https://github.com/PerformanceEstimation/Tutorial-SMAI-MODE`

Tools:

- ◇ Jupyter notebooks,
- ◇ Optimization tools/packages:
  - – convex optimization interfacing/modeling tool: CVXPY.
  - – Semidefinite (SDP) solvers: get academic MOSEK if possible (here).
- ◇ Symbolic computations:
  - – SymPy [**OK for this mini-tutorial**], or Wolfram Engine (here+jupyter).
  - – Mathematica is better (but expensive) — free (limited) version via cloud: here.

Do interrupt with questions! Pace & program probably optimistic and to be adapted.