ORF522 – Linear and Nonlinear Optimization

19. Computer-aided analysis of first-order methods

Bartolomeo Stellato — Fall 2024

Today’s lecture

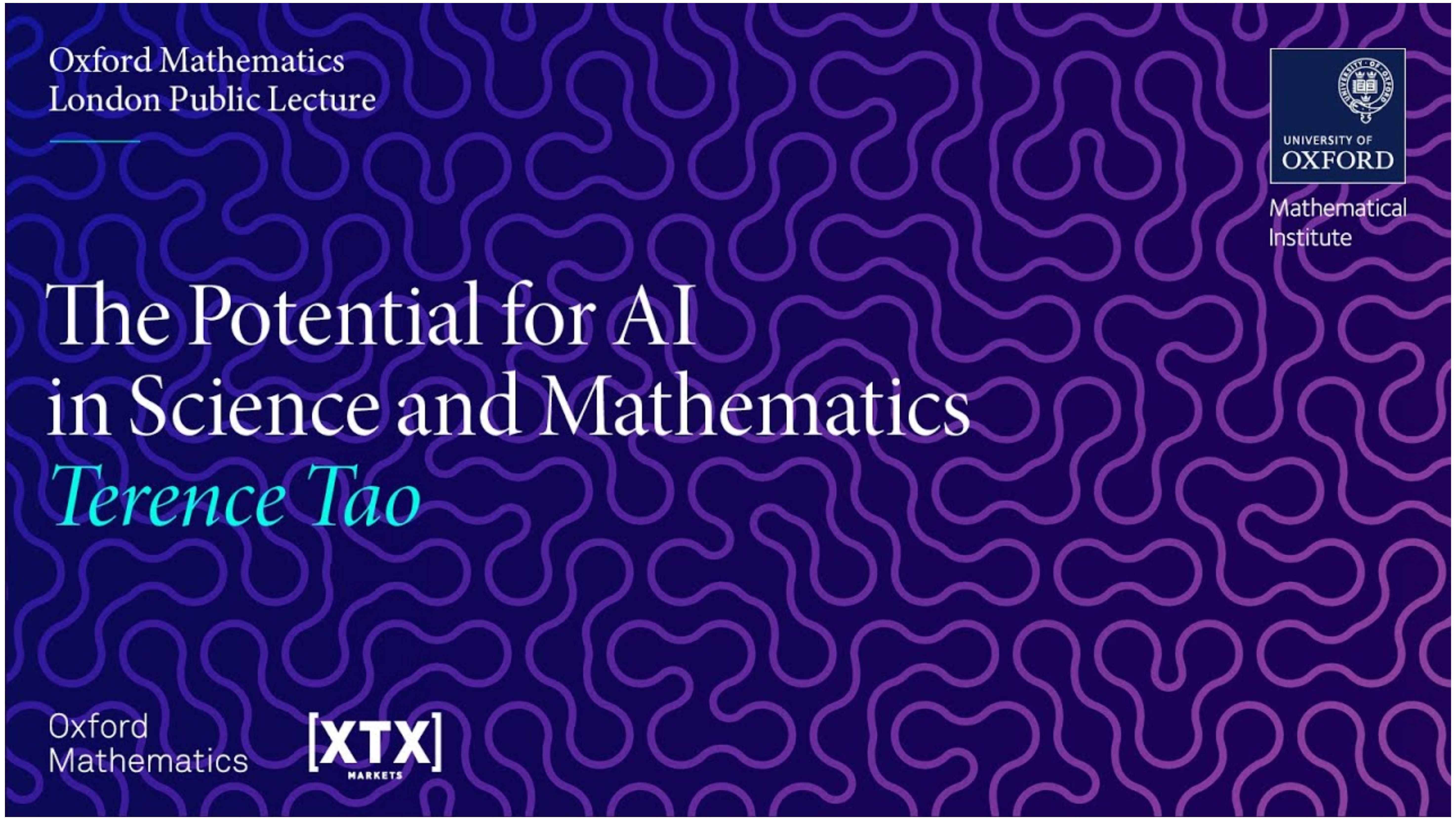
Computer Assisted Analysis and Large Scale Convex Optimization Review

- Analyzing gradient descent using computer-assisted proofs
- Performance estimation
- Summary of large-scale convex optimization

Material
- Blog post by Francis Bach: https://francisbach.com/computer-aided-analyses/
- Adrien Taylor’s tutorials https://adrientaylor.github.io/tutorials/
- Lots of exciting papers by Drori, Bach, Lessard, Hendrickx, de Klerk, Ryu, Bolte, and others….

Computer-assisted proof techniques are growing

Generative AI is a great guessing machine → It works well if we can check the correctness of the results!



https://youtu.be/_sTDSO74D8Q?si=U7IYHZB8kBDYEPxv

Lean4 is a theorem proving language
(used to check AlphaProof)

Today we will see a different technique to
analyze first-order methods!

Gradient descent example

Analysis of a gradient step

Unconstrained smooth optimization

minimize $f(x)$ $x \in \mathbf{R}^n$

under some assumptions on f

gradient descent

$$x^{k+1} = x^k - t\nabla f(x^k)$$

What guarantees we can give in terms of the following performance metrics after N iterations?

- Cost function distance: $e(x) = f(x) - f(x^\star)$
- Solution distance: $e(x) = \|x - x^\star\|$
- Gradient norm: $e(x) = \|\nabla f(x)\|$

Convergence rate of a gradient step

For error $e(x) = \|\nabla f(x)\|$, find the smallest β such that

$$\|\nabla f(x^1)\| \leq \beta \|\nabla f(x^0)\| \quad \forall x^0, x^1$$

for $x^1 = x^0 - t\nabla f(x^0)$

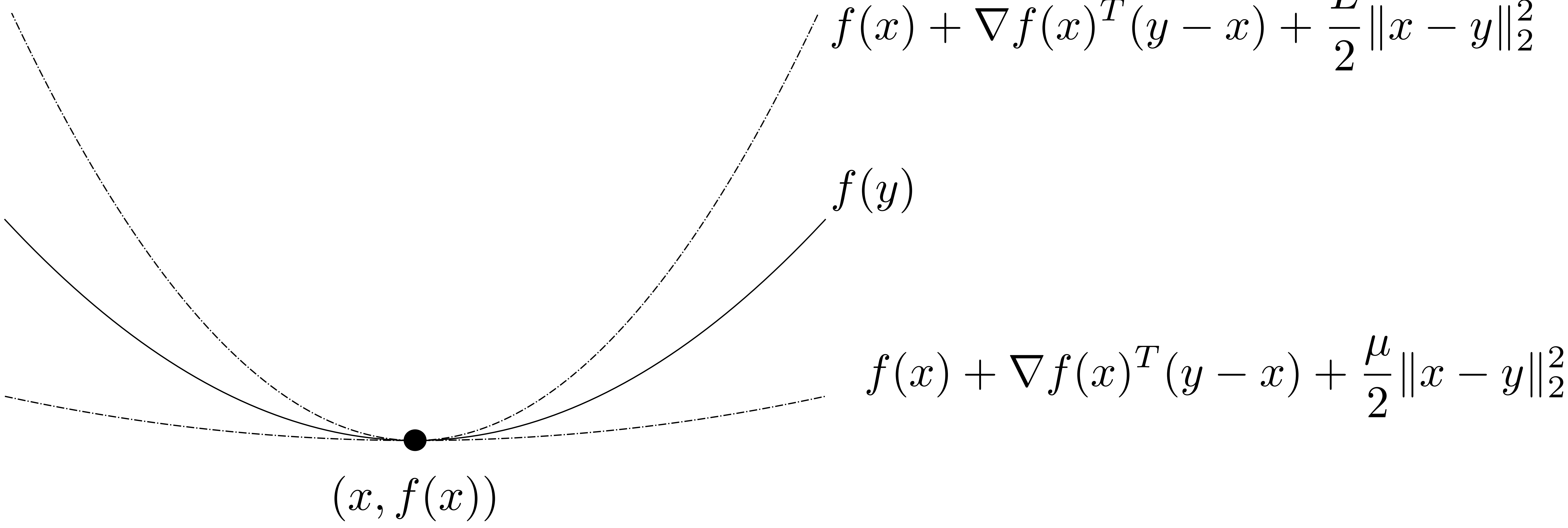
We can write it as an optimization problem

$$\begin{array}{ll}\underset{f, x^1, x^0}{\text{maximize}} & \|\nabla f(x^1)\| \\ \text{subject to} & x^1 = x^0 - t\nabla f(x^0) \\ & \text{assumptions on } f \\ & \|\nabla f(x^0)\| \leq 1\end{array}$$

We need assumptions on the problem function

L-smoothness: $$f(y) \leq f(x) + \nabla f(x)^T(y-x) + \frac{L}{2}\|x-y\|_2^2$$

μ-strong convexity: $$f(y) \geq f(x) + \nabla f(x)^T(y-x) + \frac{\mu}{2}\|x-y\|_2^2, \quad \forall x, y$$



We choose $f \in \mathcal{F}_{\mu,L}$, the class of μ-strongly convex and L-smooth functions

Back to the convergence rate problem

$$\begin{array}{ll} \underset{f, x^1, x^0}{\text{maximize}} & \|\nabla f(x^1)\| \\ \text{subject to} & x^1 = x^0 - t\nabla f(x^0) \\ & f \in \mathcal{F}_{\mu,L} \\ & \|\nabla f(x^0)\| \leq 1 \end{array}$$

(t, μ, L are problem parameters)

$f \in \mathcal{F}_{\mu,L}$ ← strongly convex and smooth functions

The theoretical worst-case value is

$$\|\nabla f(x^1)\|^2 \leq \max\{(1 - t\mu)^2, (1 - tL)^2\}\|\nabla f(x^0)\| \quad \forall x^0, x^1$$

which gives the optimal step size $t = \dfrac{2}{\mu + L}$ (from gradient descent lecture)

How can we solve the maximization problem?

From infinite to finite dimensional optimization

$$f \in \mathcal{F}_{\mu,L}$$

issues

1. f is a function (infinite dimensional variable)
2. the set $\mathcal{F}_{\mu,L}$ represents functions

idea

1. replace f by its discrete representation

$$f^0 = f(x^0), \quad g^0 = \nabla f(x^0)$$
$$f^1 = f(x^1), \quad g^1 = \nabla f(x^1)$$

2. require points (x^i, g^i, f^i) to be *interpolable* by a function $f \in \mathcal{F}_{\mu,L}$

Discretized worst-case problem

$$\begin{aligned} \underset{f, x^1, x^0}{\text{maximize}} \quad & \|\nabla f(x^1)\| \\ \text{subject to} \quad & x^1 = x^0 - t\nabla f(x^0) \\ & f \in \mathcal{F}_{\mu,L} \\ & \|\nabla f(x^0)\| \leq 1 \end{aligned}$$

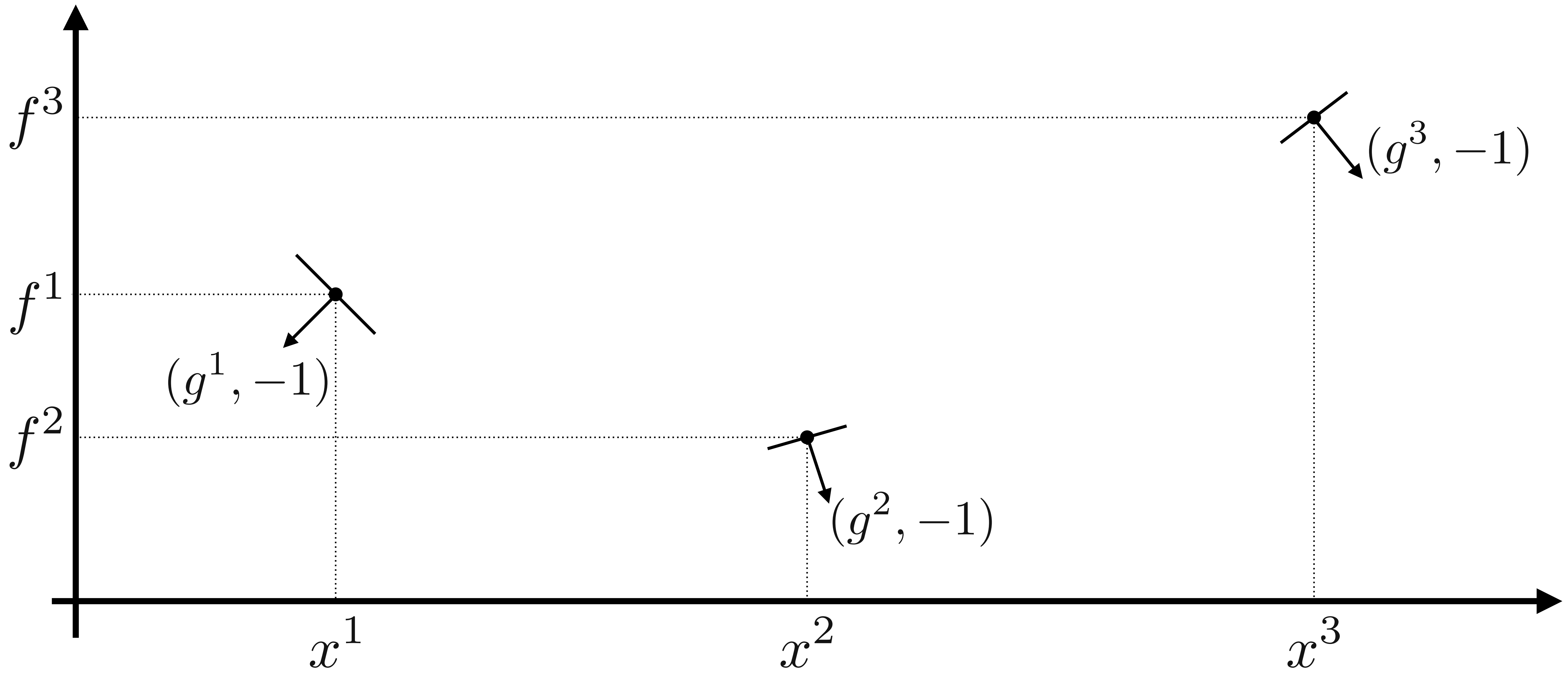
↓

$$\begin{aligned} \underset{\substack{f^1, f^0, g^1, \\ g^0, x^1, x^0}}{\text{maximize}} \quad & \|g^1\| \\ \text{subject to} \quad & x^1 = x^0 - tg^0 \\ & \exists f \in \mathcal{F}_{\mu,L} \text{ such that } \begin{cases} f^i = f(x^i) \\ g^i = \nabla f(x^i) \end{cases} \\ & \|g^0\| \leq 1 \end{aligned}$$

Smooth and strongly convex interpolation

Consider an index set I with associated tuples $\{(x^i, g^i, f^i\}_{i \in I}$

(sub)gradients



question

$\exists f \in \mathcal{F}_{\mu,L}$ such that $\begin{cases} f^i = f(x^i) \\ g^i = \nabla f(x^i) \end{cases}$ **?**

Necessary and sufficient conditions $\forall i, j \in I$

$$f^i \geq f^j + (g^j)^T(x_i - x_j) + \frac{1}{2L}\|g^i - g^j\|^2 + \frac{\mu}{2(1-\mu/L)}\left\|x^i - x^j - \frac{1}{L}(g^i - g^j)\right\|^2$$

Discretized worst-case problem with interpolation constraints

$$\begin{aligned}
\underset{\substack{f^1, f^0, g^1,\\ g^0, x^1, x^0}}{\text{maximize}} \quad & \|g^1\| \\
\text{subject to} \quad & x^1 = x^0 - tg^0 \\
& f^1 \geq f^0 + (g^0)^T(x^1 - x^0) + \tfrac{1}{2L}\|g^1 - g^0\|^2 + \tfrac{\mu}{2(1-\mu/L)}\left\|x^1 - x^0 - \tfrac{1}{L}(g^1 - g^0)\right\|^2 \\
& f^0 \geq f^1 + (g^1)^T(x^0 - x^1) + \tfrac{1}{2L}\|g^0 - g^1\|^2 + \tfrac{\mu}{2(1-\mu/L)}\left\|x^0 - x^1 - \tfrac{1}{L}(g^0 - g^1)\right\|^2 \\
& \|g^0\| \leq 1
\end{aligned}$$

$\exists f \in \mathcal{F}_{\mu,L}$ such that $\begin{cases} f^i = f(x^i) \\ g^i = \nabla f(x^i) \end{cases}$

Substitute gradient step $x^1 = x^0 - tg^0$

$$\begin{aligned}
\underset{\{(x^i, g^i, f^i)\}_{i\in\{0,1\}}}{\text{maximize}} \quad & \|g^1\| \\
\text{subject to} \quad & f^1 \geq f^0 - t\|g^0\|^2 + \tfrac{1}{2L}\|g^1 - g^0\|^2 + \tfrac{\mu}{2(1-\mu/L)}\left\|\left(\tfrac{1}{L} - t\right)g^0 - \tfrac{1}{L}g^1\right\|^2 \\
& f^0 \geq f^1 + t(g^1)^T g^0 + \tfrac{1}{2L}\|g^0 - g^1\|^2 + \tfrac{\mu}{2(1-\mu/L)}\left\|\left(t - \tfrac{1}{L}\right)g^0 + \tfrac{1}{L}g^1\right\|^2 \\
& \|g^0\| \leq 1
\end{aligned}$$

nonconvex quadratic constraints

Semidefinite programming lifting procedure

We stack variables in matrix $P = [x^0 \; x^1 \; g^0 \; g^1] \in \mathbf{R}^{n \times 4}$

Define Gram matrix

$$G = P^T P = \begin{bmatrix} (x^0)^T x^0 & (x^0)^T x^1 & (x^0)^T g^0 & (x^0)^T g^1 \\ (x^1)^T x^0 & (x^1)^T x^1 & (x^1)^T g^0 & (x^1)^T g^1 \\ (g^0)^T x^0 & (g^0)^T x^1 & (g^0)^T g^0 & (g^0)^T g^1 \\ (g^1)^T x^0 & (g^1)^T x^1 & (g^1)^T g^0 & (g^1)^T g^1 \end{bmatrix} \succeq 0$$

Our problem is **linear** in G!

$$G \succeq 0 \text{ and } \mathbf{rank}(G) \leq n \quad \Longleftrightarrow \quad G = P^T P \text{ with } P \in \mathbf{R}^{n \times 4}$$

Since $G \in \mathbf{R}^{4 \times 4}$ we have $\mathbf{rank}(G) \leq 4$ ⟶ Therefore, rank constraint disappears when $n \geq 4$

$\Rightarrow$ We can recover $P = [x^0 \; x^1 \; g^0 \; g^1]$ from G with a Cholesky factorization.

Semidefinite formulation

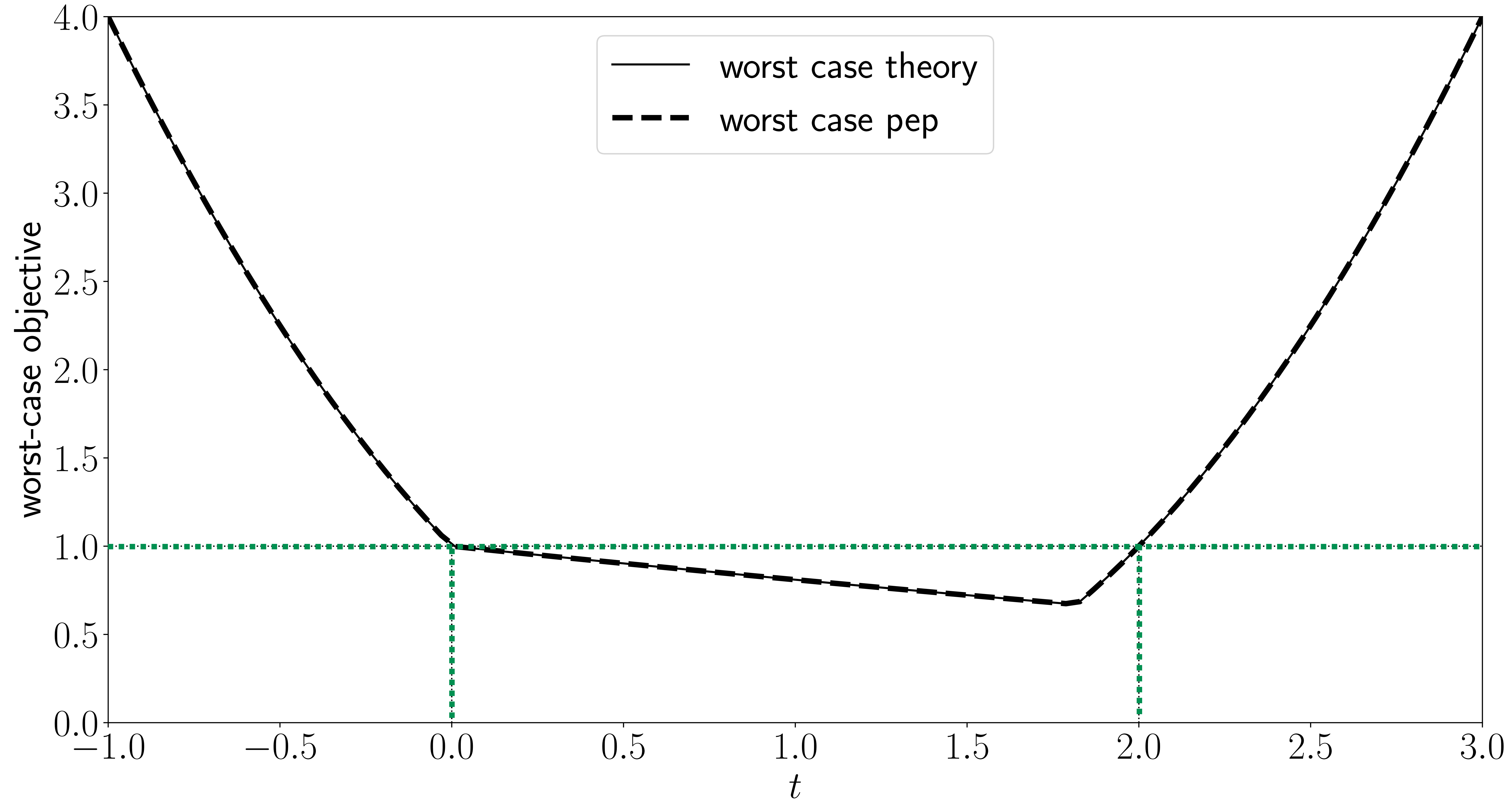
- encode objective $\|g^1\|^2 = (g^1)^T g^1 = G_{44}$
- encode initial condition $\|g^0\|^2 = (g^0)^T g^0 = G_{33} \leq 1$
- encode interpolation constraints as $f^j - f^i + \mathbf{tr}(GA_{ij}) \leq 0$ for some A_{ij}

automated performance estimation problem

$$
\begin{array}{ll}
\underset{G, f^1, f^0}{\text{maximize}} & G_{44} \\
\text{subject to} & f^j - f^i + \mathbf{tr}(GA_{ij}) \leq 0, \quad i, j \in \{0, 1\} \\
& G \succeq 0 \\
& G_{33} \leq 1
\end{array}
$$

Solving the SDP

Fix $L = 1, \mu = 0.1$ and solve SDP for varying step size t



can we translate this into analytical guarantees?

Exactly matches $\max\{(1-t\mu)^2, (1-tL)^2\}$-convergence for $t \in (0, 2/L)$

Divergence (> 1) for $t < 0$ or $t > 2$

Analytical proofs with duality

gradient step with $t = 1/L$
$((1-\mu/L)^2 = (1-t\mu)^2 \geq (1-tL)^2 = 0)$

interpolation inequalities

$$f^1 \geq f^0 + \nabla f(x^0)^T(x^1 - x^0) + \tfrac{1}{2L}\|\nabla f(x^1) - \nabla f(x^0)\|^2 + \tfrac{\mu}{2(1-\mu/L)}\left\|x^1 - x^0 - \tfrac{1}{L}(\nabla f(x^1) - \nabla f(x^0))\right\|^2$$

$$f^0 \geq f^1 + \nabla f(x^1)^T(x^0 - x^1) + \tfrac{1}{2L}\|\nabla f(x^0) - \nabla f(x^1)\|^2 + \tfrac{\mu}{2(1-\mu/L)}\left\|x^0 - x^1 - \tfrac{1}{L}(\nabla f(x^0) - \nabla f(x^1))\right\|^2$$

dual variables

$$\lambda_1 = \frac{2}{t}(1-\mu t) \geq 0$$

$$\lambda_2 = \frac{2}{t}(1-\mu t) \geq 0$$

guess *(from numerical values)*

Weighted sum of the constraints with weights λ_1, λ_2 can be written as

$$\begin{aligned}\|\nabla f(x^1)\|^2 &\leq (1-t\mu)^2\|\nabla f(x^0)\|^2 - \tfrac{2-t(L+\mu)}{t(L-\mu)}\|(1-t\mu)\nabla f(x^0) - \nabla f(x^1)\|^2 \\ &\leq (1-t\mu)^2\|\nabla f(x^0)\|^2 \\ &\leq (1-t\mu)^2\end{aligned}$$

≥ 0 ($= 0$ at the worst-case)

with $t = 1/L$ we have the convergence rate $\|\nabla f(x^1)\|^2 \leq (1-\mu/L)^2\|\nabla f(x^0)\|^2$ (tight)

Remarks on dual problem

interpretation

- find the smallest upper bound that can be proved by a linear combination of the interpolation inequalities
- we can show that strong duality holds (existence of Slater's point)
  - any convergence rate (primal objective) can be proved by a linear combination of interpolation inequalities (dual objective)
  - any dual feasible point can be translated into "traditional" (SDP-less) proofs

how to build purely analytical proofs?

- we need to "guess" how the optimal dual variables depend on problem parameters
- SDP optimal values gives us a way to check correctness

Performance estimation

Performance Estimation Problem (PEP)

Features

- any primal solution gives a lower bound (i.e., function)
- any dual solution is a worst-case guarantee (i.e., a proof)
- both can be computed using semidefinite programming (SDP)

Algorithms (with accelerated variants)

- (sub)gradient methods
- proximal point methods
- projected and proximal gradients methods
- splitting methods
- randomized/stochastic gradient methods
- distributed/decentralized gradient methods
- … and many more!

Classes of optimization problems

We can model any composite optimization problem of the form

$$\text{minimize} \quad f(x) + h(x)$$

For many functional classes in convex optimization::

- different types of (smooth or non-smooth) functions
- convex indicator and support functions
- monotone inclusion problems
- … and more

any class whose interpolation conditions are SDP-representable

Performance metrics

common errors

- Cost function distance: $e(x) = f(x) - f(x^\star)$
- Solution distance: $e(x) = \|x - x^\star\|$
- Gradient norm: $e(x) = \|\nabla f(x)\|$

best error along the way

$$\min_{0 \leq i \leq N} e(x^i)$$

any linear function of

f_i and gram matrix entries $\|x^i\|^2, \|g^i\|^2, (g^i)^T x^j$

PEPit toolbox

https://github.com/PerformanceEstimation/PEPit

- Works in Python
- Used to analyze virtually any first-order method used in convex optimization (includes stochastic, and continuous-time methods)
- Interfaces with cvxpy to call an SDP solver

```
problem = PEP()
func = problem.declare_function(
    function_class=SmoothStronglyConvexFunction, mu=mu, L=L)
x0 = problem.set_initial_point()
x1 = x0 - t * func.gradient(x0)
problem.set_initial_condition(func.gradient(x0) ** 2 <= 1)
problem.set_performance_metric(func.gradient(x1) ** 2)
worst_case_value = problem.solve()
```

Can be used to design algorithms as well

Optimized Gradient Method

$$x^{k+1} = y^k - \frac{1}{L}\nabla f(y^k)$$

$$y^{k+1} = x^{k+1} + \frac{\theta_k - 1}{\theta^{k+1}}(x^{k+1} - x^k) + \frac{\theta_k}{\theta^{k+1}}(x^{k+1} - y^k)$$

(for appropriately chosen θ_k)

tight convergence guarantee
(lower and upper bounds match exactly up to constants)

Y. Drori, M. Teboulle (2014). Performance of first-order methods for smooth convex minimization: a novel approach. Mathematical Programming
D. Kim, J. Fessler (2016). Optimized first-order methods for smooth convex minimization. Mathematical Programming
Y. Drori (2017). The exact information-based complexity of smooth convex minimization. Journal of Complexity

Numerically optimal step sizes

Solve minmax problem using branch-and-bound:

1. we minimize over step sizes t_k
2. we maximize over PEP problem variables (f^i, G,...)

S. Das Gupta, B. Van Parys, E. Ryu, (2024) "Branch-and-Bound Performance Estimation Programming: A Unified Methodology for Constructing Optimal Optimization Methods", Mathematical Programming

Many more (active research area…)

Limitations of PEP

- Results are **not interpretable** in terms of problem parameters. You need to **“guess” the connections**.

- If you already have an optimal algorithm matching lower bounds (e.g., in Nesterov acceleration), **PEP cannot give you better rates**. It can give you the exact constant in front of the rate.

- **SDPs can become very large for 50/100 steps** and take a very long time

- Results are **dimension-independent**: cannot represent exactly the iterates because they disappear in the Gram matrix

Summary of large-scale convex optimization

Large-scale convex optimization

Optimality conditions

- KKT optimality conditions
- Subgradient optimality conditions $0 \in \partial f(x^\star)$

General
Necessary

Convex
Necessary and sufficient

First order methods: Moderate accuracy on Large-scale data

- Gradient descent
- Subgradient methods
- Proximal algorithms (e.g., ISTA)
- Operator splitting algorithms (e.g., ADMM)

Convergence rates

Typical rates
(gradient descent, proximal gradient, ADMM, etc.)

- L-smoothness: $O(1/k)$, accelerated $O(1/k^2)$
- μ-strong convexity: $O(\log(1/k))$
- We can always combine **line search**
- Convergence bounds usually in terms of cost function distance

Operator theory

- Helps developing and analyzing serial and distributed algorithms
- Algorithms **always** converge for convex problems (independently from step size)
- Convergence bounds usually in terms of iterates distance

First-order methods

Per-iteration cost ↓

Number of iterations ↑

- Gradient/subgradient method
- Forward-backward splitting (proximal algorithms)
- Accelerated forward-backward splitting
- Douglas-Rachford splitting (ADMM)
- Interior-point methods (not covered)

Large-scale systems

- start with feasible method with cheapest per-iteration cost
- if too many iterations, transverse down the list

Computer-assisted analysis of optimization algorithms

Today, we learned to:

- **Formulate** performance analysis problem using semidefinite programming
- **Recover** known convergence rates by observing SDP solution
- **Prove** convergence rates using dual variables by combining interpolation inequalities
- **Select** the appropriate algorithms to apply in large-scale optimization

Next lecture

- Extensions and nonconvex and stochastic optimization