Performance Estimation Problems

A Tutorial – Session 1 / 3

March 2026

Aymeric Dieuleveut, Adrien Taylor

March 18, 2026

Outline

Outline

What is this lecture about?

Performance Estimation = computer-aided techniques for proofs in first-order optimization

Goal of the first session (today):

- Give you a working understanding of PEP

Plan:

What is this lecture about?

Performance Estimation = computer-aided techniques for proofs in first-order optimization

Goal of the first session (today):

- Give you a working understanding of PEP
- Show the step-by-step derivation on some concrete examples

Plan:

What is this lecture about?

Performance Estimation = computer-aided techniques for proofs in first-order optimization

Goal of the first session (today):

- Give you a working understanding of PEP
- Show the step-by-step derivation on some concrete examples
- Give you a broad understanding of what is doable or not

Plan:

What is this lecture about?

Performance Estimation = computer-aided techniques for proofs in first-order optimization

Goal of the first session (today):

- Give you a working understanding of PEP
- Show the step-by-step derivation on some concrete examples
- Give you a broad understanding of what is doable or not
- See how we build counter-example function
- Connect to proofs

Plan:

What is this lecture about?

Performance Estimation = computer-aided techniques for proofs in first-order optimization

Goal of the first session (today):

- Give you a working understanding of PEP
- Show the step-by-step derivation on some concrete examples
- Give you a broad understanding of what is doable or not
- See how we build counter-example function
- Connect to proofs

Plan:

1. GD reminders & convergence proof

What is this lecture about?

Performance Estimation = computer-aided techniques for proofs in first-order optimization

Goal of the first session (today):

- Give you a working understanding of PEP
- Show the step-by-step derivation on some concrete examples
- Give you a broad understanding of what is doable or not
- See how we build counter-example function
- Connect to proofs

Plan:

1. GD reminders & convergence proof
2. Primal-PEP crash course:
   - 2.1 Example 1 - $\|x_1 - x_\star\|^2$
   - 2.2 *Interactive session*

What is this lecture about?

Performance Estimation = computer-aided techniques for proofs in first-order optimization

Goal of the first session (today):

- Give you a working understanding of PEP
- Show the step-by-step derivation on some concrete examples
- Give you a broad understanding of what is doable or not
- See how we build counter-example function
- Connect to proofs

Plan:

1. GD reminders & convergence proof
2. Primal-PEP crash course:
   - 2.1 Example 1 - $\|x_1 - x_\star\|^2$
   - 2.2 *Interactive session*
   - 2.3 Example 2 - $f(x_1) - f_\star$
   - 2.4 Generic framework & PEPit

What is this lecture about?

Performance Estimation = computer-aided techniques for proofs in first-order optimization

Goal of the first session (today):

- Give you a working understanding of PEP
- Show the step-by-step derivation on some concrete examples
- Give you a broad understanding of what is doable or not
- See how we build counter-example function
- Connect to proofs

Plan:

1. GD reminders & convergence proof
2. Primal-PEP crash course:
   - 2.1 Example 1 - $\|x_1 - x_\star\|^2$
   - 2.2 *Interactive session*
   - 2.3 Example 2 - $f(x_1) - f_\star$
   - 2.4 Generic framework & PEPit
   - 2.5 Technical leftovers

What is this lecture about?

Performance Estimation = computer-aided techniques for proofs in first-order optimization

Goal of the first session (today):

- Give you a working understanding of PEP
- Show the step-by-step derivation on some concrete examples
- Give you a broad understanding of what is doable or not
- See how we build counter-example function
- Connect to proofs

Plan:

1. GD reminders & convergence proof
2. Primal-PEP crash course:
   - 2.1 Example 1 - $\|x_1 - x_\star\|^2$
   - 2.2 *Interactive session*
   - 2.3 Example 2 - $f(x_1) - f_\star$
   - 2.4 Generic framework & PEPit
   - 2.5 Technical leftovers
3. Counter-examples

What is this lecture about?

Performance Estimation = computer-aided techniques for proofs in first-order optimization

Goal of the first session (today):

- Give you a working understanding of PEP
- Show the step-by-step derivation on some concrete examples
- Give you a broad understanding of what is doable or not
- See how we build counter-example function
- Connect to proofs

Plan:

1. GD reminders & convergence proof
2. Primal-PEP crash course:
   - 2.1 Example 1 - $\|x_1 - x_\star\|^2$
   - 2.2 *Interactive session*
   - 2.3 Example 2 - $f(x_1) - f_\star$
   - 2.4 Generic framework & PEPit
   - 2.5 Technical leftovers
3. Counter-examples
4. Duality and proofs

Motivation: why do we need tight convergence analyses?

Motivation: why do we need tight convergence analyses?

→ First-order optimization is a domain with many proofs with structure.

→ Traditional analyses are:
- error-prone & technical
- lack global insights

✘ Error-prone — ? Easily fixable?

✘ Technical — ? Simple to adapt?

✘ Few patterns — ? Algo variations?

Motivation: why do we need tight convergence analyses?

→ First-order optimization is a domain with many proofs with structure.

→ Traditional analyses are:
- error-prone & technical
- lack global insights
- hard to adapt to algorithmic variations

✘ Error-prone ? Easily fixable?

✘ Technical ? Simple to adapt?

✘ Few patterns ? Algo variations?

Motivation: why do we need tight convergence analyses?

→ First-order optimization is a domain with many proofs with structure.

→ Traditional analyses are:
- error-prone & technical
- lack global insights
- hard to adapt to algorithmic variations

→ **Take away**: for most first-order algorithms and simple function classes, we can get the tightest convergence rate numerically.

✘ Error-prone ? Easily fixable?

✘ Technical ? Simple to adapt?

✘ Few patterns ? Algo variations?

Setting: first-order optimization

Minimize $f : \mathbb{R}^d \to \mathbb{R}$:

$$x_\star = \arg\min_{x \in \mathbb{R}^d} f(x),$$

where $f \in \mathcal{F}$ (encodes assumptions on the function class).

Setting: first-order optimization

Minimize $f : \mathbb{R}^d \to \mathbb{R}$:

$$x_\star = \arg\min_{x \in \mathbb{R}^d} f(x),$$

where $f \in \mathcal{F}$ (encodes assumptions on the function class).

Use a simple first-order algorithm, e.g. gradient descent:

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

Setting: first-order optimization

Minimize $f : \mathbb{R}^d \to \mathbb{R}$:

$$x_\star = \arg\min_{x \in \mathbb{R}^d} f(x),$$

where $f \in \mathcal{F}$ (encodes assumptions on the function class).

Use a simple first-order algorithm, e.g. gradient descent:

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

Performance guarantees? Bounds on

$$\frac{f(x_k) - f(x_\star)}{\|x_0 - x_\star\|^2}, \quad \frac{\|x_k - x_\star\|^2}{\|x_0 - x_\star\|^2}, \quad \frac{\|\nabla f(x_k)\|^2}{f(x_0) - f(x_\star)}, \quad f(x_k) - f(x_\star), \ldots$$

Setting: first-order optimization

Minimize $f : \mathbb{R}^d \to \mathbb{R}$:

$$x_\star = \arg\min_{x \in \mathbb{R}^d} f(x),$$

where $f \in \mathcal{F}$ (encodes assumptions on the function class).

Use a simple first-order algorithm, e.g. gradient descent:

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

Performance guarantees? Bounds on

$$\frac{f(x_k) - f(x_\star)}{\|x_0 - x_\star\|^2}, \quad \frac{\|x_k - x_\star\|^2}{\|x_0 - x_\star\|^2}, \quad \frac{\|\nabla f(x_k)\|^2}{f(x_0) - f(x_\star)}, \quad f(x_k) - f(x_\star), \ldots$$

Outline

Regularity conditions: Smoothness

Definition : L-Smoothness

A differentiable function f is L-smooth if ∇f is L-Lipschitz:

$$\|\nabla f(x) - \nabla f(y)\| \leq L\|x - y\| \quad \forall x, y \in \mathbb{R}^d.$$

Regularity conditions: Smoothness

Definition : L-Smoothness

A differentiable function f is L-smooth if ∇f is L-Lipschitz:

$$\|\nabla f(x) - \nabla f(y)\| \leq L\|x - y\| \quad \forall x, y \in \mathbb{R}^d.$$

Proposition : Smooth + Convex characterization

A differentiable f is L-smooth and convex if and only if for all x, y:

$$f(y) + \langle \nabla f(y), x - y \rangle \;\leq\; f(x) \;\leq\; f(y) + \langle \nabla f(y), x - y \rangle + \tfrac{L}{2}\|x - y\|^2.$$

Regularity conditions: Smoothness

Definition : L-Smoothness

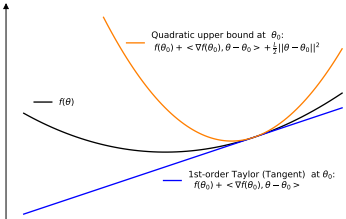
A differentiable function f is L-smooth if ∇f is L-Lipschitz:

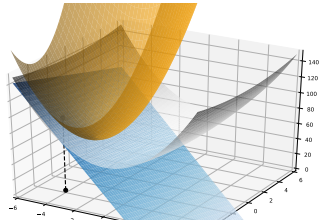
$$\|\nabla f(x) - \nabla f(y)\| \leq L\|x - y\| \quad \forall x, y \in \mathbb{R}^d.$$

Proposition : Smooth + Convex characterization

A differentiable f is L-smooth and convex if and only if for all x, y:

$$f(y) + \langle \nabla f(y), x - y \rangle \;\leq\; f(x) \;\leq\; f(y) + \langle \nabla f(y), x - y \rangle + \frac{L}{2}\|x - y\|^2.$$





Observing f and ∇f *at* y provides information on the function *everywhere*.

Regularity conditions: Strong Convexity & the class $\mathcal{F}_{\mu,L}$

Definition : μ-Strong Convexity

f is μ-strongly convex if for all $x, y \in \mathbb{R}^d$:

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

Regularity conditions: Strong Convexity & the class $\mathcal{F}_{\mu,L}$

Definition : μ-Strong Convexity

f is μ-strongly convex if for all $x, y \in \mathbb{R}^d$:

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

Definition : Function class $\mathcal{F}_{\mu,L}$

$\mathcal{F}_{\mu,L}$ denotes the set of μ-strongly convex and L-smooth functions on $\mathbb{R}^d$. (We allow $\mu = 0$ for convex-only.)

Regularity conditions: Strong Convexity & the class $\mathcal{F}_{\mu,L}$

Definition : μ-Strong Convexity

f is μ-strongly convex if for all $x, y \in \mathbb{R}^d$:

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

Definition : Function class $\mathcal{F}_{\mu,L}$

$\mathcal{F}_{\mu,L}$ denotes the set of μ-strongly convex and L-smooth functions on $\mathbb{R}^d$. (We allow $\mu = 0$ for convex-only.)

Proposition : Co-coercivity (key inequality for GD analysis)

If $f \in \mathcal{F}_{\mu,L}$, then for all x, y:

$$\langle \nabla f(x) - \nabla f(y),\, x - y \rangle \;\geq\; \frac{\mu L}{\mu + L}\|x - y\|^2 + \frac{1}{\mu + L}\|\nabla f(x) - \nabla f(y)\|^2.$$

Gradient Descent: convergence rate

Theorem : Convergence of GD on $\mathcal{F}_{\mu,L}$

Let $f \in \mathcal{F}_{\mu,L}$ with $0 \leq \mu \leq L$.

GD with step size $\eta = \dfrac{2}{\mu + L}$ satisfies, after t iterations:

Gradient Descent: convergence rate

Theorem : Convergence of GD on $\mathcal{F}_{\mu,L}$

Let $f \in \mathcal{F}_{\mu,L}$ with $0 \leq \mu \leq L$.

GD with step size $\eta = \dfrac{2}{\mu + L}$ satisfies, after t iterations:

$$\|x_t - x_\star\|^2 \leq \left(\frac{L-\mu}{L+\mu}\right)^{2t} \|x_0 - x_\star\|^2.$$

Gradient Descent: convergence rate

Theorem : Convergence of GD on $\mathcal{F}_{\mu,L}$

Let $f \in \mathcal{F}_{\mu,L}$ with $0 \leq \mu \leq L$.

GD with step size $\eta = \dfrac{2}{\mu + L}$ satisfies, after t iterations:

$$\|x_t - x_\star\|^2 \leq \left(\frac{L-\mu}{L+\mu}\right)^{2t} \|x_0 - x_\star\|^2.$$

$\rightarrow$ For $\mu = 0$: convergence rate is 1 (no linear rate), but $f(x_t) - f_\star \leq \dfrac{L\|x_0 - x_\star\|^2}{2t}$.

Gradient Descent: convergence rate

Theorem : Convergence of GD on $\mathcal{F}_{\mu,L}$

Let $f \in \mathcal{F}_{\mu,L}$ with $0 \leq \mu \leq L$.

GD with step size $\eta = \dfrac{2}{\mu + L}$ satisfies, after t iterations:

$$\|x_t - x_\star\|^2 \leq \left(\frac{L-\mu}{L+\mu}\right)^{2t} \|x_0 - x_\star\|^2.$$

→ For $\mu = 0$: convergence rate is 1 (no linear rate), but $f(x_t) - f_\star \leq \dfrac{L\|x_0 - x_\star\|^2}{2t}$.

→ The rate $\left(\dfrac{L-\mu}{L+\mu}\right)^2 \approx 1 - \dfrac{2\mu}{L}$ for $\kappa = L/\mu \gg 1$.

Proof of convergence (sketch)

Step 1. Expand the squared distance:

$$\|x_{t+1} - x_\star\|^2 = \|x_t - \eta \nabla f(x_t) - x_\star\|^2 = \|x_t - x_\star\|^2 - 2\eta \langle x_t - x_\star, \nabla f(x_t) \rangle + \eta^2 \|\nabla f(x_t)\|^2.$$

Proof of convergence (sketch)

Step 1. Expand the squared distance:

$$\|x_{t+1} - x_\star\|^2 = \|x_t - \eta \nabla f(x_t) - x_\star\|^2 = \|x_t - x_\star\|^2 - 2\eta \langle x_t - x_\star,\, \nabla f(x_t) \rangle + \eta^2 \|\nabla f(x_t)\|^2.$$

Step 2. Apply co-coercivity (with $\nabla f(x_\star) = 0$):

$$\langle \nabla f(x_t),\, x_t - x_\star \rangle \;\geq\; \frac{\mu L}{\mu + L} \|x_t - x_\star\|^2 + \frac{1}{\mu + L} \|\nabla f(x_t)\|^2.$$

Proof of convergence (sketch)

Step 1. Expand the squared distance:

$$\|x_{t+1} - x_\star\|^2 = \|x_t - \eta \nabla f(x_t) - x_\star\|^2 = \|x_t - x_\star\|^2 - 2\eta \langle x_t - x_\star,\, \nabla f(x_t) \rangle + \eta^2 \|\nabla f(x_t)\|^2.$$

Step 2. Apply co-coercivity (with $\nabla f(x_\star) = 0$):

$$\langle \nabla f(x_t),\, x_t - x_\star \rangle \;\geq\; \frac{\mu L}{\mu + L} \|x_t - x_\star\|^2 + \frac{1}{\mu + L} \|\nabla f(x_t)\|^2.$$

Step 3. Plug in and collect:

$$\|x_{t+1} - x_\star\|^2 \;\leq\; \left(1 - \frac{2\eta\mu L}{\mu + L}\right) \|x_t - x_\star\|^2 + \left(\eta^2 - \frac{2\eta}{\mu + L}\right) \|\nabla f(x_t)\|^2.$$

Proof of convergence (sketch)

Step 1. Expand the squared distance:

$$\|x_{t+1} - x_\star\|^2 = \|x_t - \eta \nabla f(x_t) - x_\star\|^2 = \|x_t - x_\star\|^2 - 2\eta \langle x_t - x_\star,\, \nabla f(x_t) \rangle + \eta^2 \|\nabla f(x_t)\|^2.$$

Step 2. Apply co-coercivity (with $\nabla f(x_\star) = 0$):

$$\langle \nabla f(x_t),\, x_t - x_\star \rangle \;\geq\; \frac{\mu L}{\mu + L} \|x_t - x_\star\|^2 + \frac{1}{\mu + L} \|\nabla f(x_t)\|^2.$$

Step 3. Plug in and collect:

$$\|x_{t+1} - x_\star\|^2 \;\leq\; \left(1 - \frac{2\eta\mu L}{\mu + L}\right) \|x_t - x_\star\|^2 + \left(\eta^2 - \frac{2\eta}{\mu + L}\right) \|\nabla f(x_t)\|^2.$$

Step 4. Choose $\eta = \dfrac{2}{\mu + L}$: the gradient term vanishes and

$$1 - \frac{2\eta\mu L}{\mu + L} = \frac{(L - \mu)^2}{(L + \mu)^2}.$$

Proof of convergence (sketch)

Step 1. Expand the squared distance:

$$\|x_{t+1} - x_\star\|^2 = \|x_t - \eta \nabla f(x_t) - x_\star\|^2 = \|x_t - x_\star\|^2 - 2\eta \langle x_t - x_\star,\, \nabla f(x_t)\rangle + \eta^2 \|\nabla f(x_t)\|^2.$$

Step 2. Apply co-coercivity (with $\nabla f(x_\star) = 0$):

$$\langle \nabla f(x_t),\, x_t - x_\star \rangle \;\geq\; \frac{\mu L}{\mu + L} \|x_t - x_\star\|^2 + \frac{1}{\mu + L} \|\nabla f(x_t)\|^2.$$

Step 3. Plug in and collect:

$$\|x_{t+1} - x_\star\|^2 \;\leq\; \left(1 - \frac{2\eta\mu L}{\mu + L}\right) \|x_t - x_\star\|^2 + \left(\eta^2 - \frac{2\eta}{\mu + L}\right) \|\nabla f(x_t)\|^2.$$

Step 4. Choose $\eta = \dfrac{2}{\mu + L}$: the gradient term vanishes and

$$1 - \frac{2\eta\mu L}{\mu + L} = \frac{(L - \mu)^2}{(L + \mu)^2}.$$

$$\Rightarrow \text{By induction: } \|x_t - x_\star\|^2 \leq \left(\frac{L - \mu}{L + \mu}\right)^{2t} \|x_0 - x_\star\|^2.$$

Outline

EXAMPLE ONE

—

Primal PEP for squared distance

Crash course on Performance Estimation Problems

→ **Performance Estimation problems**: rethinking proofs of first-order optimization [1].

What is a worst-case guarantee (WCG)?

Example:

$$\forall f \in \mathcal{F}_{\mu,L}, \quad \text{for } x_1 = x_0 - L^{-1}\nabla f(x_0): \quad \|x_1 - x_\star\|^2 \leq \tau \|x_0 - x_\star\|^2$$

$\forall f \in \mathcal{F}_{\mu,L}$, for $x_1 = x_0 - L^{-1}\nabla f(x_0)$: $\|x_1 - x_\star\|^2 \leq \tau \|x_0 - x_\star\|^2$

$$\forall f \in \mathcal{F}_{\mu,L}, \quad \text{for } x_1 = x_0 - L^{-1}\nabla f(x_0): \quad \|x_1 - x_\star\|^2 \leq \tau \|x_0 - x_\star\|^2$$

$$\tau^* = \sup_{f,\, x_0} \frac{\|x_1 - x_\star\|^2}{\|x_0 - x_\star\|^2},$$

$$\forall f \in \mathcal{F}_{\mu,L}, \quad \text{for } x_1 = x_0 - L^{-1}\nabla f(x_0): \quad \|x_1 - x_\star\|^2 \leq \tau \|x_0 - x_\star\|^2$$

$$\tau^* = \sup_{f,\, x_0} \frac{\|x_1 - x_\star\|^2}{\|x_0 - x_\star\|^2}, \quad \textbf{s.t.} \quad f \in \mathcal{F}_{\mu,L}, \ \textbf{with } x_1 = x_0 - L^{-1}\nabla f(x_0), \ \ x_\star = \arg\min f$$

Crash course on Performance Estimation Problems

$$\forall f \in \mathcal{F}_{\mu,L}, \quad \text{for } x_1 = x_0 - L^{-1}\nabla f(x_0): \quad \|x_1 - x_\star\|^2 \leq \tau \|x_0 - x_\star\|^2$$

$$\tau^* = \sup_{f, x_0, x_1, x_\star} \frac{\|x_1 - x_\star\|^2}{\|x_0 - x_\star\|^2}, \quad \textbf{s.t.} \quad f \in \mathcal{F}_{\mu,L}, \ \textbf{and } x_1 = x_0 - L^{-1}\nabla f(x_0), \ \ x_\star = \arg\min f$$

$$\forall f \in \mathcal{F}_{\mu,L}, \quad \text{for } x_1 = x_0 - L^{-1}\nabla f(x_0): \quad \|x_1 - x_\star\|^2 \leq \tau \|x_0 - x_\star\|^2$$

$$\begin{aligned}
\tau^* &= \sup_{f,\, x_0,\, x_1,\, x_\star} \frac{\|x_1 - x_\star\|^2}{\|x_0 - x_\star\|^2}, \quad \textbf{s.t.} \quad f \in \mathcal{F}_{\mu,L}, \ \textbf{and } x_1 = x_0 - L^{-1}\nabla f(x_0), \ \ x_\star = \arg\min f \\
&= \sup_{f,\, x_0, x_1, x_\star, g_0} \frac{\|x_1 - x_\star\|^2}{\|x_0 - x_\star\|^2}, \quad \textbf{s.t.} \quad f \in \mathcal{F}_{\mu,L}, \ x_1 = x_0 - L^{-1} g_0, \ x_\star = \arg\min f \\
&\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad g_0 = \nabla f(x_0),
\end{aligned}$$

$$\forall f \in \mathcal{F}_{\mu,L}, \quad \text{for } x_1 = x_0 - L^{-1}\nabla f(x_0): \quad \|x_1 - x_\star\|^2 \leq \tau \|x_0 - x_\star\|^2$$

$$\begin{aligned}
\tau^* &= \sup_{f,\, x_0,\, x_1,\, x_\star} \frac{\|x_1 - x_\star\|^2}{\|x_0 - x_\star\|^2}, \quad \textbf{s.t.} \quad f \in \mathcal{F}_{\mu,L}, \ \textbf{and } x_1 = x_0 - L^{-1}\nabla f(x_0), \ \ x_\star = \arg\min f \\
&= \sup_{f,\, x_0, x_1, x_\star, g_0} \frac{\|x_1 - x_\star\|^2}{\|x_0 - x_\star\|^2}, \quad \textbf{s.t.} \quad f \in \mathcal{F}_{\mu,L}, \ x_1 = x_0 - L^{-1} g_0, \ x_\star = \arg\min f \\
&\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad g_0 = \nabla f(x_0), \ x_\star = 0,
\end{aligned}$$

$$\forall f \in \mathcal{F}_{\mu,L}, \quad \text{for } x_1 = x_0 - L^{-1}\nabla f(x_0): \quad \|x_1 - x_\star\|^2 \leq \tau \|x_0 - x_\star\|^2$$

$$\begin{aligned}
\tau^* &= \sup_{f,\, x_0,\, x_1,\, x_\star} \frac{\|x_1 - x_\star\|^2}{\|x_0 - x_\star\|^2}, \quad \textbf{s.t.} \quad f \in \mathcal{F}_{\mu,L}, \ \textbf{and } x_1 = x_0 - L^{-1}\nabla f(x_0), \ x_\star = \arg\min f \\
&= \sup_{f,\, x_0, x_1, x_\star, g_0} \frac{\|x_1\|^2}{\|x_0\|^2}, \quad \textbf{s.t.} \quad f \in \mathcal{F}_{\mu,L}, \ x_1 = x_0 - L^{-1} g_0, \ x_\star = \arg\min f \\
&\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad g_0 = \nabla f(x_0), \ x_\star = 0,
\end{aligned}$$

$$\forall f \in \mathcal{F}_{\mu,L}, \quad \text{for } x_1 = x_0 - L^{-1}\nabla f(x_0): \quad \|x_1 - x_\star\|^2 \leq \tau \|x_0 - x_\star\|^2$$

$$\begin{aligned}
\tau^* &= \sup_{f,\, x_0,\, x_1,\, x_\star} \frac{\|x_1 - x_\star\|^2}{\|x_0 - x_\star\|^2}, \quad \textbf{s.t.} \quad f \in \mathcal{F}_{\mu,L}, \ \textbf{and } x_1 = x_0 - L^{-1}\nabla f(x_0), \ x_\star = \arg\min f \\
&= \sup_{f,\, x_0, x_1, x_\star, g_0} \frac{\|x_1\|^2}{\|x_0\|^2}, \quad \textbf{s.t.} \quad f \in \mathcal{F}_{\mu,L}, \ x_1 = x_0 - L^{-1} g_0, \ x_\star = \arg\min f \\
&\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad g_0 = \nabla f(x_0), \ x_\star = 0, \ \|x_0\|^2 = 1
\end{aligned}$$

$$\forall f \in \mathcal{F}_{\mu,L}, \quad \text{for } x_1 = x_0 - L^{-1}\nabla f(x_0): \quad \|x_1 - x_\star\|^2 \leq \tau \|x_0 - x_\star\|^2$$

$$\begin{aligned}
\tau^* &= \sup_{f,\, x_0,\, x_1,\, x_\star} \frac{\|x_1 - x_\star\|^2}{\|x_0 - x_\star\|^2}, \quad \textbf{s.t.} \quad f \in \mathcal{F}_{\mu,L}, \ \textbf{and } x_1 = x_0 - L^{-1}\nabla f(x_0), \ x_\star = \arg\min f \\
&= \sup_{f,\, x_0, x_1, x_\star, g_0} \|x_1\|^2, \quad \textbf{s.t.} \quad f \in \mathcal{F}_{\mu,L}, \ x_1 = x_0 - L^{-1} g_0, \ x_\star = \arg\min f \\
&\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad g_0 = \nabla f(x_0), \ x_\star = 0, \ \|x_0\|^2 = 1
\end{aligned}$$

$$\forall f \in \mathcal{F}_{\mu,L}, \quad \text{for } x_1 = x_0 - L^{-1}\nabla f(x_0): \quad \|x_1 - x_\star\|^2 \leq \tau \|x_0 - x_\star\|^2$$

$$\begin{aligned}
\tau^* &= \sup_{f,\, x_0,\, x_1,\, x_\star} \frac{\|x_1 - x_\star\|^2}{\|x_0 - x_\star\|^2}, \quad \textbf{s.t.} \quad f \in \mathcal{F}_{\mu,L}, \ \textbf{and } x_1 = x_0 - L^{-1}\nabla f(x_0), \ x_\star = \arg\min f \\
&= \sup_{f,\, x_0, x_1, x_\star, g_0} \|x_1\|^2, \quad \textbf{s.t.} \quad f \in \mathcal{F}_{\mu,L}, \ x_1 = x_0 - L^{-1} g_0, \ x_\star = \arg\min f \\
&\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad g_0 = \nabla f(x_0), \ x_\star = 0, \ \|x_0\|^2 = 1 \\
&= \sup_{x_0, x_1, x_\star, g_0} \|x_1\|^2, \quad \textbf{s.t.} \quad x_1 = x_0 - L^{-1} g_0, \ x_\star = 0, \ \|x_0\|^2 = 1
\end{aligned}$$

$$\forall f \in \mathcal{F}_{\mu,L}, \quad \text{for } x_1 = x_0 - L^{-1}\nabla f(x_0): \quad \|x_1 - x_\star\|^2 \leq \tau \|x_0 - x_\star\|^2$$

$$\begin{aligned}
\tau^* &= \sup_{f,\, x_0,\, x_1,\, x_\star} \frac{\|x_1 - x_\star\|^2}{\|x_0 - x_\star\|^2}, \quad \textbf{s.t.} \quad f \in \mathcal{F}_{\mu,L}, \ \textbf{and } x_1 = x_0 - L^{-1}\nabla f(x_0), \ x_\star = \arg\min f \\
&= \sup_{f,\, x_0, x_1, x_\star, g_0} \|x_1\|^2, \quad \textbf{s.t.} \quad f \in \mathcal{F}_{\mu,L}, \ x_1 = x_0 - L^{-1} g_0, \ x_\star = \arg\min f \\
&\qquad\qquad g_0 = \nabla f(x_0), \ x_\star = 0, \ \|x_0\|^2 = 1 \\
&= \sup_{x_0, x_1, x_\star, g_0} \|x_1\|^2, \quad \textbf{s.t.} \quad x_1 = x_0 - L^{-1} g_0, \ x_\star = 0, \ \|x_0\|^2 = 1 \\
&\qquad\qquad \exists f \in \mathcal{F}_{\mu,L} : \ g_0 = \nabla f(x_0), \ \nabla f(x_\star) = 0
\end{aligned}$$

$$\forall f \in \mathcal{F}_{\mu,L}, \quad \text{for } x_1 = x_0 - L^{-1}\nabla f(x_0): \quad \|x_1 - x_\star\|^2 \leq \tau \|x_0 - x_\star\|^2$$

$$\begin{aligned}
\tau^* &= \sup_{f,\, x_0,\, x_1,\, x_\star} \frac{\|x_1 - x_\star\|^2}{\|x_0 - x_\star\|^2}, \quad \textbf{s.t.} \quad f \in \mathcal{F}_{\mu,L}, \ \textbf{and } x_1 = x_0 - L^{-1}\nabla f(x_0), \ x_\star = \arg\min f \\
&= \sup_{f,\, x_0, x_1, x_\star, g_0} \|x_1\|^2 \quad , \quad \textbf{s.t.} \quad f \in \mathcal{F}_{\mu,L}, \ x_1 = x_0 - L^{-1} g_0, \ x_\star = \arg\min f \\
&\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad g_0 = \nabla f(x_0), \ x_\star = 0, \ \|x_0\|^2 = 1 \\
&= \sup_{x_0, x_1, x_\star, g_0} \|x_1\|^2, \quad \textbf{s.t.} \quad x_1 = x_0 - L^{-1} g_0, \ x_\star = 0, \ \|x_0\|^2 = 1 \\
&\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad \exists f \in \mathcal{F}_{\mu,L} : \ g_0 = \nabla f(x_0), \ \nabla f(x_\star) = 0 \\
&= \sup_{x_0, x_1, g_0} \|x_1\|^2, \quad \textbf{s.t.} \quad x_1 = x_0 - L^{-1} g_0, \ x_\star = 0, \|x_0\|^2 = 1
\end{aligned}$$

$$\forall f \in \mathcal{F}_{\mu,L}, \quad \text{for } x_1 = x_0 - L^{-1}\nabla f(x_0): \quad \|x_1 - x_\star\|^2 \leq \tau \|x_0 - x_\star\|^2$$

$$\begin{aligned}
\tau^* &= \sup_{f,\, x_0,\, x_1,\, x_\star} \frac{\|x_1 - x_\star\|^2}{\|x_0 - x_\star\|^2}, \quad \textbf{s.t.} \quad f \in \mathcal{F}_{\mu,L}, \ \textbf{and } x_1 = x_0 - L^{-1}\nabla f(x_0), \ x_\star = \arg\min f \\
&= \sup_{f,\, x_0, x_1, x_\star, g_0} \|x_1\|^2 \quad , \quad \textbf{s.t.} \quad f \in \mathcal{F}_{\mu,L}, \ x_1 = x_0 - L^{-1} g_0, \ x_\star = \arg\min f \\
&\qquad\qquad g_0 = \nabla f(x_0), \ x_\star = 0, \ \|x_0\|^2 = 1 \\
&= \sup_{x_0, x_1, x_\star, g_0} \|x_1\|^2, \quad \textbf{s.t.} \quad x_1 = x_0 - L^{-1} g_0, \ x_\star = 0, \ \|x_0\|^2 = 1 \\
&\qquad\qquad \exists f \in \mathcal{F}_{\mu,L} : \ g_0 = \nabla f(x_0), \ \nabla f(x_\star) = 0 \\
&= \sup_{x_0, x_1, g_0} \|x_1\|^2, \quad \textbf{s.t.} \quad x_1 = x_0 - L^{-1} g_0, \ x_\star = 0, \|x_0\|^2 = 1 \\
&\qquad\qquad \langle g_0, x_0 \rangle \geq \tfrac{1}{L}\|g_0\|^2 + \tfrac{\mu}{1-\mu/L}\left\|x_0 - \tfrac{1}{L} g_0\right\|^2.
\end{aligned}$$

$$\forall f \in \mathcal{F}_{\mu,L}, \quad \text{for } x_1 = x_0 - L^{-1}\nabla f(x_0): \quad \|x_1 - x_\star\|^2 \leq \tau \|x_0 - x_\star\|^2$$

$$\tau^* = \sup_{x_0, x_1, g_0} \|x_1\|^2, \quad \textbf{s.t.} \quad x_1 = x_0 - L^{-1} g_0, \ \|x_0\|^2 = 1$$

$$\langle g_0, x_0 \rangle \geq \tfrac{1}{L}\|g_0\|^2 + \tfrac{\mu}{1-\mu/L}\left\|x_0 - \tfrac{1}{L}g_0\right\|^2.$$

$$\forall f \in \mathcal{F}_{\mu,L}, \quad \text{for } x_1 = x_0 - L^{-1}\nabla f(x_0): \quad \|x_1 - x_\star\|^2 \leq \tau \|x_0 - x_\star\|^2$$

$$\tau^* = \sup_{x_0, x_1, g_0} \|x_1\|^2, \quad \textbf{s.t.} \quad x_1 = x_0 - L^{-1} g_0, \ \|x_0\|^2 = 1$$

$$\langle g_0, x_0 \rangle \geq \tfrac{1}{L}\|g_0\|^2 + \tfrac{\mu}{1-\mu/L}\left\|x_0 - \tfrac{1}{L}g_0\right\|^2.$$

$$G = \begin{pmatrix} x_0 \\ x_1 \\ g_0 \end{pmatrix} \begin{pmatrix} x_0 \\ x_1 \\ g_0 \end{pmatrix}^\top = \begin{pmatrix} \|x_0\|^2 & \langle x_0, x_1 \rangle & \langle x_0, g_0 \rangle \\ \langle x_0, x_1 \rangle & \|x_1\|^2 & \langle x_1, g_0 \rangle \\ \langle x_0, g_0 \rangle & \langle x_1, g_0 \rangle & \|g_0\|^2 \end{pmatrix} \succcurlyeq 0$$

$$\forall f \in \mathcal{F}_{\mu,L}, \quad \text{for } x_1 = x_0 - L^{-1}\nabla f(x_0): \quad \|x_1 - x_\star\|^2 \leq \tau \|x_0 - x_\star\|^2$$

$$\begin{aligned}
\tau^* &= \sup_{x_0, x_1, g_0} \quad \|x_1\|^2, \quad \textbf{s.t.} \quad x_1 = x_0 - L^{-1} g_0, \ \|x_0\|^2 = 1 \\
&\qquad\qquad \langle g_0, x_0\rangle \geq \tfrac{1}{L}\|g_0\|^2 + \tfrac{\mu}{1-\mu/L}\left\|x_0 - \tfrac{1}{L}g_0\right\|^2. \\
&= \sup_{0 \preccurlyeq G} \quad G_{2,2}, \quad \textbf{s.t.} \quad \mathrm{Tr}(G\, C_{\text{alg}}) = 0, \ G_{1,1} = 1, \quad \mathrm{Tr}(G\, C_{\text{class}}) \geq 0.
\end{aligned}$$

$$G = \begin{pmatrix} x_0 \\ x_1 \\ g_0 \end{pmatrix} \begin{pmatrix} x_0 \\ x_1 \\ g_0 \end{pmatrix}^\top = \begin{pmatrix} \|x_0\|^2 & \langle x_0, x_1\rangle & \langle x_0, g_0\rangle \\ \langle x_0, x_1\rangle & \|x_1\|^2 & \langle x_1, g_0\rangle \\ \langle x_0, g_0\rangle & \langle x_1, g_0\rangle & \|g_0\|^2 \end{pmatrix} \succcurlyeq 0$$

$$\forall f \in \mathcal{F}_{\mu,L}, \quad \text{for } x_1 = x_0 - L^{-1}\nabla f(x_0): \quad \|x_1 - x_\star\|^2 \leq \tau \|x_0 - x_\star\|^2$$

$$\begin{aligned}
\tau^* &= \sup_{x_0, x_1, g_0} \quad \|x_1\|^2, \quad \textbf{s.t.} \quad x_1 = x_0 - L^{-1} g_0, \ \|x_0\|^2 = 1 \\
&\qquad\qquad \langle g_0, x_0 \rangle \geq \tfrac{1}{L}\|g_0\|^2 + \tfrac{\mu}{1-\mu/L}\left\|x_0 - \tfrac{1}{L} g_0\right\|^2. \\
&= \sup_{0 \preccurlyeq G} \quad G_{2,2}, \quad \textbf{s.t.} \quad \mathrm{Tr}(G\, C_{\text{alg}}) = 0, \ G_{1,1} = 1, \quad \mathrm{Tr}(G\, C_{\text{class}}) \geq 0.
\end{aligned}$$

$$G = \begin{pmatrix} x_0 \\ x_1 \\ g_0 \end{pmatrix} \begin{pmatrix} x_0 \\ x_1 \\ g_0 \end{pmatrix}^\top = \begin{pmatrix} \|x_0\|^2 & \langle x_0, x_1 \rangle & \langle x_0, g_0 \rangle \\ \langle x_0, x_1 \rangle & \|x_1\|^2 & \langle x_1, g_0 \rangle \\ \langle x_0, g_0 \rangle & \langle x_1, g_0 \rangle & \|g_0\|^2 \end{pmatrix} \succcurlyeq 0$$

Key insight The problem has been lifted to a semidefinite program (SDP) – a convex problem!

What did we just do? – Summary

$$\forall f \in \mathcal{F}_{\mu,L}, \quad \text{for } x_1 = x_0 - L^{-1}\nabla f(x_0): \quad \|x_1 - x_\star\|^2 \leq \tau \|x_0 - x_\star\|^2$$

Step 1. Write the tightest τ as a supremum:

$$\tau^* = \sup_{\substack{f \in \mathcal{F}_{\mu,L},\ x_0, x_\star \\ x_1 = x_0 - L^{-1}\nabla f(x_0),\ x_\star = \arg\min f}} \frac{\|x_1 - x_\star\|^2}{\|x_0 - x_\star\|^2}$$

What did we just do? – Summary

$$\forall f \in \mathcal{F}_{\mu,L}, \quad \text{for } x_1 = x_0 - L^{-1}\nabla f(x_0): \quad \|x_1 - x_\star\|^2 \leq \tau \|x_0 - x_\star\|^2$$

Step 1. Write the tightest τ as a supremum:

$$\tau^* = \sup_{\substack{f \in \mathcal{F}_{\mu,L},\ x_0, x_\star \\ x_1 = x_0 - L^{-1}\nabla f(x_0),\ x_\star = \arg\min f}} \frac{\|x_1 - x_\star\|^2}{\|x_0 - x_\star\|^2}$$

Steps 2–3. Reduce to a finite dimensional problem (no more f):

- Sample $g_0 = \nabla f(x_0)$; set $x_\star = 0$ and $\|x_0\|^2 = 1$ by homogeneity.
- Replace $\exists f \in \mathcal{F}_{\mu,L}$ by **interpolation conditions** (Taylor et al. '17).

$$\begin{aligned} \tau^* = \sup_{x_0, x_1, g_0} \quad & \|x_1\|^2 \\ \textbf{s.t.} \quad & x_1 = x_0 - L^{-1} g_0, \quad \|x_0\|^2 = 1 \\ & \langle g_0, x_0 \rangle \geq \tfrac{1}{L}\|g_0\|^2 + \tfrac{\mu}{1-\mu/L}\left\|x_0 - \tfrac{g_0}{L}\right\|^2. \end{aligned}$$

What did we just do? – Summary

$$\forall f \in \mathcal{F}_{\mu,L}, \quad \text{for } x_1 = x_0 - L^{-1}\nabla f(x_0): \quad \|x_1 - x_\star\|^2 \leq \tau \|x_0 - x_\star\|^2$$

Step 1. Write the tightest τ as a supremum:

$$\tau^* = \sup_{\substack{f \in \mathcal{F}_{\mu,L},\ x_0, x_\star \\ x_1 = x_0 - L^{-1}\nabla f(x_0),\ x_\star = \arg\min f}} \frac{\|x_1 - x_\star\|^2}{\|x_0 - x_\star\|^2}$$

Steps 2–3. Reduce to a finite dimensional problem (no more f):

- Sample $g_0 = \nabla f(x_0)$; set $x_\star = 0$ and $\|x_0\|^2 = 1$ by homogeneity.
- Replace $\exists f \in \mathcal{F}_{\mu,L}$ by **interpolation conditions** (Taylor et al. '17).

$$\begin{aligned} \tau^* = \sup_{x_0, x_1, g_0} \quad & \|x_1\|^2 \\ \textbf{s.t.} \quad & x_1 = x_0 - L^{-1} g_0, \quad \|x_0\|^2 = 1 \\ & \langle g_0, x_0 \rangle \geq \tfrac{1}{L}\|g_0\|^2 + \tfrac{\mu}{1-\mu/L}\left\|x_0 - \tfrac{g_0}{L}\right\|^2. \end{aligned}$$

Key point: the problem is now finite-dimensional (variables $x_0, x_1, g_0 \in \mathbb{R}^d$) and quadratic in (x_0, g_0).
$\rightsquigarrow$ One more step: lift to a convex SDP.

Step 4: lifting to a semidefinite program

Step 4. Every quadratic in (x_0, x_1, g_0) is **linear** in the Gram matrix $G = vv^\top$, $v = (x_0, x_1, g_0)$:

$$G = \begin{pmatrix} x_0 \\ x_1 \\ g_0 \end{pmatrix} \begin{pmatrix} x_0 \\ x_1 \\ g_0 \end{pmatrix}^\top = \begin{pmatrix} \|x_0\|^2 & \langle x_0, x_1 \rangle & \langle x_0, g_0 \rangle \\ \langle x_1, x_0 \rangle & \|x_1\|^2 & \langle x_1, g_0 \rangle \\ \langle g_0, x_0 \rangle & \langle g_0, x_1 \rangle & \|g_0\|^2 \end{pmatrix} \succcurlyeq 0.$$

Step 4: lifting to a semidefinite program

Step 4. Every quadratic in (x_0, x_1, g_0) is **linear** in the Gram matrix $G = vv^\top$, $v = (x_0, x_1, g_0)$:

$$G = \begin{pmatrix} x_0 \\ x_1 \\ g_0 \end{pmatrix} \begin{pmatrix} x_0 \\ x_1 \\ g_0 \end{pmatrix}^\top = \begin{pmatrix} \|x_0\|^2 & \langle x_0, x_1 \rangle & \langle x_0, g_0 \rangle \\ \langle x_1, x_0 \rangle & \|x_1\|^2 & \langle x_1, g_0 \rangle \\ \langle g_0, x_0 \rangle & \langle g_0, x_1 \rangle & \|g_0\|^2 \end{pmatrix} \succcurlyeq 0.$$

The PEP becomes a standard SDP:

$$\begin{aligned} \tau^* = \quad & \sup_{G \succcurlyeq 0} \quad G_{22} \\ & \mathrm{Tr}(G\, C_{\mathrm{alg}}) = 0 \\ & G_{11} = 1 \\ & \mathrm{Tr}(G\, C_{\mathrm{class}}) \geq 0 \end{aligned}$$

Step 4: lifting to a semidefinite program

Step 4. Every quadratic in (x_0, x_1, g_0) is **linear** in the Gram matrix $G = vv^\top$, $v = (x_0, x_1, g_0)$:

$$G = \begin{pmatrix} x_0 \\ x_1 \\ g_0 \end{pmatrix} \begin{pmatrix} x_0 \\ x_1 \\ g_0 \end{pmatrix}^\top = \begin{pmatrix} \|x_0\|^2 & \langle x_0, x_1 \rangle & \langle x_0, g_0 \rangle \\ \langle x_1, x_0 \rangle & \|x_1\|^2 & \langle x_1, g_0 \rangle \\ \langle g_0, x_0 \rangle & \langle g_0, x_1 \rangle & \|g_0\|^2 \end{pmatrix} \succcurlyeq 0.$$

The PEP becomes a standard SDP:

$$\begin{aligned} \tau^* = \quad & \sup_{G \succcurlyeq 0} \quad G_{22} \\ & \mathrm{Tr}(G\, C_{\mathrm{alg}}) = 0 \\ & G_{11} = 1 \\ & \mathrm{Tr}(G\, C_{\mathrm{class}}) \geq 0 \end{aligned}$$

Basic take-away

→ **Convex SDP**: solvable with any solver.

Interactive Session #1

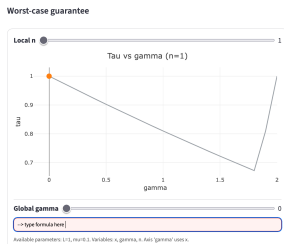
Exploring PEP numerically with the Streamlit UI

Activity: GD on L-smooth and μ-strongly convex functions — what is the worst-case rate for $\|x_1 - x_\star\|^2$?

1. Go to `pepitui-test.streamlit.app`
2. Select **Gradient Descent** (preimplemented), use *Customize* to set the performance metric to $\|x_1 - x_\star\|^2$ vs $\|x_0 - x_\star\|^2$
3. Set the function class to $\mathcal{F}_{\mu,L}$ (smooth **and** strongly convex)
4. Set $\mu = 0.1$, $L = 1$
5. Run `plot` and observe the worst-case value τ^*

$\Rightarrow$ **Try to guess the formula for** τ^* as a function of μ and L.

$\Rightarrow$ **Which step size gives the best rate?**



Outline

EXAMPLE TWO

—

Primal PEP for function-value criterion

PEP for $f(x_1) - f_\star \leq \tau \|x_0 - x_\star\|^2$

$\forall f \in \mathcal{F}_{0,L}$, for $x_1 = x_0 - \frac{1}{L}\nabla f(x_0)$: $f(x_1) - f_\star \leq \tau \|x_0 - x_\star\|^2$

PEP for $f(x_1) - f_\star \leq \tau \|x_0 - x_\star\|^2$

$\forall f \in \mathcal{F}_{0,L}$, for $x_1 = x_0 - \frac{1}{L}\nabla f(x_0)$: $f(x_1) - f_\star \leq \tau \|x_0 - x_\star\|^2$

$$\tau^* = \sup_{f, x_0, x_1, x_\star} \frac{f(x_1) - f_\star}{\|x_0 - x_\star\|^2},$$

PEP for $f(x_1) - f_\star \leq \tau \|x_0 - x_\star\|^2$

$$\forall f \in \mathcal{F}_{0,L}, \quad \text{for } x_1 = x_0 - \tfrac{1}{L}\nabla f(x_0): \quad f(x_1) - f_\star \;\leq\; \tau \|x_0 - x_\star\|^2$$

$$\tau^* \;=\; \sup_{f,\, x_0,\, x_1,\, x_\star} \frac{f(x_1) - f_\star}{\|x_0 - x_\star\|^2}, \text{ s.t. } f \in \mathcal{F}_{0,L}, \text{ and } x_1 = x_0 - \tfrac{1}{L}\nabla f(x_0),\ x_\star = \arg\min f$$

PEP for $f(x_1) - f_\star \leq \tau \|x_0 - x_\star\|^2$

$$\forall f \in \mathcal{F}_{0,L}, \quad \text{for } x_1 = x_0 - \tfrac{1}{L}\nabla f(x_0): \quad f(x_1) - f_\star \;\leq\; \tau \|x_0 - x_\star\|^2$$

$$\begin{aligned}
\tau^* &= \sup_{f, x_0, x_1, x_\star} \frac{f(x_1) - f_\star}{\|x_0 - x_\star\|^2}, \text{ s.t. } f \in \mathcal{F}_{0,L}, \text{ and } x_1 = x_0 - \tfrac{1}{L}\nabla f(x_0),\ x_\star = \arg\min f \\
&= \sup_{f, x_0, x_1, x_\star, g_0, g_1, f_0, f_1, f_\star} \frac{f_1 - f_\star}{\|x_0 - x_\star\|^2}, \text{ s.t. } f \in \mathcal{F}_{0,L}, x_1 = x_0 - \tfrac{1}{L} g_0, \nabla f(x_\star) = 0 \\
&\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad g_0 = \nabla f(x_0),\ g_1 = \nabla f(x_1),\ f_i = f(x_i),\ f_\star = f(x_\star)
\end{aligned}$$

PEP for $f(x_1) - f_\star \leq \tau \|x_0 - x_\star\|^2$

$$\forall f \in \mathcal{F}_{0,L}, \quad \text{for } x_1 = x_0 - \tfrac{1}{L}\nabla f(x_0): \quad f(x_1) - f_\star \;\leq\; \tau \|x_0 - x_\star\|^2$$

$$\begin{aligned}
\tau^* &= \sup_{f,\, x_0,\, x_1,\, x_\star} \frac{f(x_1) - f_\star}{\|x_0 - x_\star\|^2}, \ \textbf{s.t. } f \in \mathcal{F}_{0,L}, \ \textbf{and } x_1 = x_0 - \tfrac{1}{L}\nabla f(x_0), \ x_\star = \arg\min f \\
&= \sup_{f,\, x_0, x_1, x_\star, g_0, g_1, f_0, f_1, f_\star} \frac{f_1 - f_\star}{\|x_0 - x_\star\|^2}, \ \textbf{s.t. } f \in \mathcal{F}_{0,L}, x_1 = x_0 - \tfrac{1}{L} g_0, \nabla f(x_\star) = 0 \\
&\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad g_0 = \nabla f(x_0), \ g_1 = \nabla f(x_1), \ f_i = f(x_i), \ f_\star = f(x_\star) \\
&\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad \|x_0 - x_\star\|^2 = 1,
\end{aligned}$$

PEP for $f(x_1) - f_\star \leq \tau \|x_0 - x_\star\|^2$

$$\forall f \in \mathcal{F}_{0,L}, \quad \text{for } x_1 = x_0 - \tfrac{1}{L}\nabla f(x_0): \quad f(x_1) - f_\star \;\leq\; \tau \|x_0 - x_\star\|^2$$

$$\begin{aligned}
\tau^* &= \sup_{f,\, x_0,\, x_1,\, x_\star} \frac{f(x_1) - f_\star}{\|x_0 - x_\star\|^2}, \text{ s.t. } f \in \mathcal{F}_{0,L}, \text{ and } x_1 = x_0 - \tfrac{1}{L}\nabla f(x_0),\ x_\star = \arg\min f \\
&= \sup_{f,\, x_0, x_1, x_\star, g_0, g_1, f_0, f_1, f_\star} f_1 - f_\star \quad , \text{ s.t. } f \in \mathcal{F}_{0,L}, x_1 = x_0 - \tfrac{1}{L} g_0, \nabla f(x_\star) = 0 \\
&\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad g_0 = \nabla f(x_0),\ g_1 = \nabla f(x_1),\ f_i = f(x_i),\ f_\star = f(x_\star) \\
&\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad \|x_0 - x_\star\|^2 = 1,
\end{aligned}$$

PEP for $f(x_1) - f_\star \leq \tau \|x_0 - x_\star\|^2$

$$\forall f \in \mathcal{F}_{0,L}, \quad \text{for } x_1 = x_0 - \tfrac{1}{L}\nabla f(x_0): \quad f(x_1) - f_\star \;\leq\; \tau \|x_0 - x_\star\|^2$$

$$\begin{aligned}
\tau^* &= \sup_{f,\, x_0,\, x_1,\, x_\star} \frac{f(x_1) - f_\star}{\|x_0 - x_\star\|^2}, \text{ s.t. } f \in \mathcal{F}_{0,L}, \text{ and } x_1 = x_0 - \tfrac{1}{L}\nabla f(x_0),\ x_\star = \arg\min f \\
&= \sup_{f,\, x_0, x_1, x_\star, g_0, g_1, f_0, f_1, f_\star} f_1 - f_\star \quad , \text{ s.t. } f \in \mathcal{F}_{0,L}, x_1 = x_0 - \tfrac{1}{L} g_0, \nabla f(x_\star) = 0 \\
&\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad g_0 = \nabla f(x_0),\ g_1 = \nabla f(x_1),\ f_i = f(x_i),\ f_\star = f(x_\star) \\
&\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad \|x_0 - x_\star\|^2 = 1, \\
&= \sup_{x_0, x_1, x_\star, g_0, g_1, f_0, f_1} f_1 - f_\star \\
&\qquad\qquad \text{s.t.} \quad x_1 = x_0 - \tfrac{1}{L} g_0,\ \|x_0 - x_\star\|^2 = 1 \\
&\qquad\qquad\qquad\quad \exists f \in \mathcal{F}_{0,L} :\ g_0 = \nabla f(x_0),\ g_1 = \nabla f(x_1), \\
&\qquad\qquad\qquad\quad f_0 = f(x_0),\ f_1 = f(x_1),\ f_\star = f(x_\star),\ \nabla f(x_\star) = 0
\end{aligned}$$

PEP for $f(x_1) - f_\star$: interpolation and SDP lifting

Three points: x_0, $x_1 = x_0 - \frac{1}{L} g_0$, $x_\star$ ($g_\star = 0$). Interpolation on $\mathcal{F}_{0,L}$: $\forall\, i, j$:

$$f_i \geq f_j + \langle g_j, x_i - x_j \rangle + \tfrac{1}{2L} \|g_i - g_j\|^2.$$

PEP for $f(x_1) - f_\star$: interpolation and SDP lifting

Three points: x_0, $x_1 = x_0 - \frac{1}{L} g_0$, $x_\star$ ($g_\star = 0$). Interpolation on $\mathcal{F}_{0,L}$: $\forall\, i, j$:

$$f_i \geq f_j + \langle g_j, x_i - x_j \rangle + \tfrac{1}{2L} \|g_i - g_j\|^2.$$

SDP lifting. Gram matrix $G \succcurlyeq 0$ and function-value vector F:

$$G = \begin{pmatrix} x_0 - x_\star \\ x_1 - x_\star \\ g_0 \\ g_1 \end{pmatrix} \begin{pmatrix} x_0 - x_\star \\ x_1 - x_\star \\ g_0 \\ g_1 \end{pmatrix}^\top \succcurlyeq 0,\ G \in \mathbb{R}^{4\times 4}, \qquad F = \begin{pmatrix} f_0 \\ f_1 \\ f_\star \end{pmatrix} \in \mathbb{R}^3.$$

Each condition: $\mathrm{Tr}(G\, M_{ij}) + F^\top v_{ij} \geq 0$.

PEP for $f(x_1) - f_\star$: interpolation and SDP lifting

Three points: x_0, $x_1 = x_0 - \frac{1}{L} g_0$, $x_\star$ ($g_\star = 0$). Interpolation on $\mathcal{F}_{0,L}$: $\forall\, i, j$:

$$f_i \geq f_j + \langle g_j, x_i - x_j \rangle + \tfrac{1}{2L} \|g_i - g_j\|^2.$$

SDP lifting. Gram matrix $G \succcurlyeq 0$ and function-value vector F:

$$G = \begin{pmatrix} x_0 - x_\star \\ x_1 - x_\star \\ g_0 \\ g_1 \end{pmatrix} \begin{pmatrix} x_0 - x_\star \\ x_1 - x_\star \\ g_0 \\ g_1 \end{pmatrix}^\top \succcurlyeq 0,\ G \in \mathbb{R}^{4\times 4}, \qquad F = \begin{pmatrix} f_0 \\ f_1 \\ f_\star \end{pmatrix} \in \mathbb{R}^3.$$

Each condition: $\mathrm{Tr}(G\, M_{ij}) + F^\top v_{ij} \geq 0$.

$$\tau^* = \sup_{\substack{0 \preccurlyeq G,\ F \in \mathbb{R}^3 \\ \mathrm{Tr}(G\, M_{\mathrm{init}}) = 1 \\ \mathrm{Tr}(G\, C_{\mathrm{alg}}) = 0 \\ \mathrm{Tr}(G\, M_{ij}) + F^\top v_{ij} \geq 0}} F^\top v_{\mathrm{perf}}$$

PEP for $f(x_1) - f_\star$: interpolation and SDP lifting

Three points: x_0, $x_1 = x_0 - \frac{1}{L} g_0$, $x_\star$ ($g_\star = 0$). Interpolation on $\mathcal{F}_{0,L}$: $\forall\, i, j$:

$$f_i \geq f_j + \langle g_j, x_i - x_j \rangle + \tfrac{1}{2L} \|g_i - g_j\|^2.$$

SDP lifting. Gram matrix $G \succcurlyeq 0$ and function-value vector F:

$$G = \begin{pmatrix} x_0 - x_\star \\ x_1 - x_\star \\ g_0 \\ g_1 \end{pmatrix} \begin{pmatrix} x_0 - x_\star \\ x_1 - x_\star \\ g_0 \\ g_1 \end{pmatrix}^\top \succcurlyeq 0,\ G \in \mathbb{R}^{4\times 4}, \qquad F = \begin{pmatrix} f_0 \\ f_1 \\ f_\star \end{pmatrix} \in \mathbb{R}^3.$$

Each condition: $\mathrm{Tr}(G\, M_{ij}) + F^\top v_{ij} \geq 0$.

$$\tau^* = \sup_{\substack{0 \preccurlyeq G,\ F \in \mathbb{R}^3 \\ \mathrm{Tr}(G\, M_{\mathrm{init}}) = 1 \\ \mathrm{Tr}(G\, C_{\mathrm{alg}}) = 0 \\ \mathrm{Tr}(G\, M_{ij}) + F^\top v_{ij} \geq 0}} F^\top v_{\mathrm{perf}}$$

(i,j)	Condition
$(\star, 0)$	$f_\star \geq f_0 + \langle g_0, x_\star - x_0 \rangle + \frac{\|g_0\|^2}{2L}$
$(0, \star)$	$f_0 \geq f_\star + \frac{\|g_0\|^2}{2L}$
$(\star, 1)$	$f_\star \geq f_1 + \langle g_1, x_\star - x_1 \rangle + \frac{\|g_1\|^2}{2L}$
$(1, \star)$	$f_1 \geq f_\star + \frac{\|g_1\|^2}{2L}$
$(0, 1)$	$f_0 \geq f_1 + \langle g_1, x_0 - x_1 \rangle + \frac{\|g_0 - g_1\|^2}{2L}$
$(1, 0)$	$f_1 \geq f_0 + \langle g_0, x_1 - x_0 \rangle + \frac{\|g_1 - g_0\|^2}{2L}$

$(\star, 0)$: $\mathrm{Tr}(G\, M_{\star 0}) + F^\top v_{\star 0} \geq 0$

$$M_{\star 0} = \begin{pmatrix} 0 & 0 & -\frac{1}{2} & 0 \\ 0 & 0 & 0 & 0 \\ -\frac{1}{2} & 0 & \frac{1}{2L} & 0 \\ 0 & 0 & 0 & 0 \end{pmatrix}$$

$$v_{\star 0} = (-1, 0, +1)^\top$$

PEP for $f(x_1) - f_\star$: interpolation and SDP lifting

Three points: $x_0,\ x_1 = x_0 - \frac{1}{L}g_0,\ x_\star\ (g_\star = 0)$. Interpolation on $\mathcal{F}_{0,L}$: $\forall\, i, j$:

$$f_i \geq f_j + \langle g_j, x_i - x_j\rangle + \tfrac{1}{2L}\|g_i - g_j\|^2.$$

SDP lifting. Gram matrix $G \succcurlyeq 0$ and function-value vector F:

$$G = \begin{pmatrix} x_0 - x_\star \\ x_1 - x_\star \\ g_0 \\ g_1 \end{pmatrix} \begin{pmatrix} x_0 - x_\star \\ x_1 - x_\star \\ g_0 \\ g_1 \end{pmatrix}^\top \succcurlyeq 0,\ G \in \mathbb{R}^{4\times 4}, \qquad F = \begin{pmatrix} f_0 \\ f_1 \\ f_\star \end{pmatrix} \in \mathbb{R}^3.$$

Each condition: $\mathrm{Tr}(G\, M_{ij}) + F^\top v_{ij} \geq 0$.

$$\tau^* = \sup_{0 \preccurlyeq G,\ F \in \mathbb{R}^3} F^\top v_{\text{perf}}$$
$$\mathrm{Tr}(G\, M_{\text{init}}) = 1$$
$$\mathrm{Tr}(G\, C_{\text{alg}}) = 0$$
$$\mathrm{Tr}(G\, M_{ij}) + F^\top v_{ij} \geq 0$$

(i,j)	Condition
$(\star, 0)$	$f_\star \geq f_0 + \langle g_0, x_\star - x_0\rangle + \frac{\|g_0\|^2}{2L}$
$(0, \star)$	$f_0 \geq f_\star + \frac{\|g_0\|^2}{2L}$
$(\star, 1)$	$f_\star \geq f_1 + \langle g_1, x_\star - x_1\rangle + \frac{\|g_1\|^2}{2L}$
$(1, \star)$	$f_1 \geq f_\star + \frac{\|g_1\|^2}{2L}$
$(0, 1)$	$f_0 \geq f_1 + \langle g_1, x_0 - x_1\rangle + \frac{\|g_0 - g_1\|^2}{2L}$
$(1, 0)$	$f_1 \geq f_0 + \langle g_0, x_1 - x_0\rangle + \frac{\|g_1 - g_0\|^2}{2L}$

$(0, \star)$: $\mathrm{Tr}(G\, M_{0\star}) + F^\top v_{0\star} \geq 0$

$$M_{0\star} = \begin{pmatrix} 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & -\frac{1}{2L} & 0 \\ 0 & 0 & 0 & 0 \end{pmatrix}$$

$$v_{0\star} = (+1, 0, -1)^\top$$

PEP for $f(x_1) - f_\star$: interpolation and SDP lifting

Three points: x_0, $x_1 = x_0 - \frac{1}{L} g_0$, $x_\star$ ($g_\star = 0$). Interpolation on $\mathcal{F}_{0,L}$: $\forall\, i, j$:

$$f_i \geq f_j + \langle g_j, x_i - x_j \rangle + \tfrac{1}{2L} \|g_i - g_j\|^2.$$

SDP lifting. Gram matrix $G \succcurlyeq 0$ and function-value vector F:

$$G = \begin{pmatrix} x_0 - x_\star \\ x_1 - x_\star \\ g_0 \\ g_1 \end{pmatrix} \begin{pmatrix} x_0 - x_\star \\ x_1 - x_\star \\ g_0 \\ g_1 \end{pmatrix}^\top \succcurlyeq 0,\ G \in \mathbb{R}^{4\times 4}, \qquad F = \begin{pmatrix} f_0 \\ f_1 \\ f_\star \end{pmatrix} \in \mathbb{R}^3.$$

Each condition: $\mathrm{Tr}(G\, M_{ij}) + F^\top v_{ij} \geq 0$.

$$\tau^* = \sup_{\substack{0 \preccurlyeq G,\ F \in \mathbb{R}^3 \\ \mathrm{Tr}(G\, M_{\mathrm{init}}) = 1 \\ \mathrm{Tr}(G\, C_{\mathrm{alg}}) = 0 \\ \mathrm{Tr}(G\, M_{ij}) + F^\top v_{ij} \geq 0}} F^\top v_{\mathrm{perf}}$$

(i,j)	Condition
$(\star, 0)$	$f_\star \geq f_0 + \langle g_0, x_\star - x_0 \rangle + \frac{\|g_0\|^2}{2L}$
$(0, \star)$	$f_0 \geq f_\star + \frac{\|g_0\|^2}{2L}$
$(\star, 1)$	$f_\star \geq f_1 + \langle g_1, x_\star - x_1 \rangle + \frac{\|g_1\|^2}{2L}$
$(1, \star)$	$f_1 \geq f_\star + \frac{\|g_1\|^2}{2L}$
$(0, 1)$	$f_0 \geq f_1 + \langle g_1, x_0 - x_1 \rangle + \frac{\|g_0 - g_1\|^2}{2L}$
$(1, 0)$	$f_1 \geq f_0 + \langle g_0, x_1 - x_0 \rangle + \frac{\|g_1 - g_0\|^2}{2L}$

$(\star, 1)$: $\mathrm{Tr}(G\, M_{\star 1}) + F^\top v_{\star 1} \geq 0$

$$M_{\star 1} = \begin{pmatrix} 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & -\frac{1}{2} \\ 0 & 0 & 0 & 0 \\ 0 & -\frac{1}{2} & 0 & \frac{1}{2L} \end{pmatrix}$$

$$v_{\star 1} = (0, -1, +1)^\top$$

PEP for $f(x_1) - f_\star$: interpolation and SDP lifting

Three points: x_0, $x_1 = x_0 - \frac{1}{L} g_0$, $x_\star$ ($g_\star = 0$). Interpolation on $\mathcal{F}_{0,L}$: $\forall\, i, j$:

$$f_i \geq f_j + \langle g_j, x_i - x_j \rangle + \tfrac{1}{2L} \|g_i - g_j\|^2.$$

SDP lifting. Gram matrix $G \succcurlyeq 0$ and function-value vector F:

$$G = \begin{pmatrix} x_0 - x_\star \\ x_1 - x_\star \\ g_0 \\ g_1 \end{pmatrix} \begin{pmatrix} x_0 - x_\star \\ x_1 - x_\star \\ g_0 \\ g_1 \end{pmatrix}^\top \succcurlyeq 0,\ G \in \mathbb{R}^{4\times 4}, \qquad F = \begin{pmatrix} f_0 \\ f_1 \\ f_\star \end{pmatrix} \in \mathbb{R}^3.$$

Each condition: $\mathrm{Tr}(G\, M_{ij}) + F^\top v_{ij} \geq 0$.

$$\tau^* = \sup_{\substack{0 \preccurlyeq G,\ F \in \mathbb{R}^3 \\ \mathrm{Tr}(G\, M_{\mathrm{init}}) = 1 \\ \mathrm{Tr}(G\, C_{\mathrm{alg}}) = 0 \\ \mathrm{Tr}(G\, M_{ij}) + F^\top v_{ij} \geq 0}} F^\top v_{\mathrm{perf}}$$

(i,j)	Condition
$(\star, 0)$	$f_\star \geq f_0 + \langle g_0, x_\star - x_0 \rangle + \frac{\|g_0\|^2}{2L}$
$(0, \star)$	$f_0 \geq f_\star + \frac{\|g_0\|^2}{2L}$
$(\star, 1)$	$f_\star \geq f_1 + \langle g_1, x_\star - x_1 \rangle + \frac{\|g_1\|^2}{2L}$
$(1, \star)$	$f_1 \geq f_\star + \frac{\|g_1\|^2}{2L}$
$(0, 1)$	$f_0 \geq f_1 + \langle g_1, x_0 - x_1 \rangle + \frac{\|g_0 - g_1\|^2}{2L}$
$(1, 0)$	$f_1 \geq f_0 + \langle g_0, x_1 - x_0 \rangle + \frac{\|g_1 - g_0\|^2}{2L}$

$(1, \star)$: $\mathrm{Tr}(G\, M_{1\star}) + F^\top v_{1\star} \geq 0$

$$M_{1\star} = \begin{pmatrix} 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & -\frac{1}{2L} \end{pmatrix}$$

$$v_{1\star} = (0, +1, -1)^\top$$

PEP for $f(x_1) - f_\star$: interpolation and SDP lifting

Three points: x_0, $x_1 = x_0 - \frac{1}{L}g_0$, $x_\star$ ($g_\star = 0$). Interpolation on $\mathcal{F}_{0,L}$: $\forall\, i, j$:

$$f_i \geq f_j + \langle g_j, x_i - x_j\rangle + \tfrac{1}{2L}\|g_i - g_j\|^2.$$

SDP lifting. Gram matrix $G \succcurlyeq 0$ and function-value vector F:

$$G = \begin{pmatrix} x_0 - x_\star \\ x_1 - x_\star \\ g_0 \\ g_1 \end{pmatrix} \begin{pmatrix} x_0 - x_\star \\ x_1 - x_\star \\ g_0 \\ g_1 \end{pmatrix}^\top \succcurlyeq 0,\ G \in \mathbb{R}^{4\times 4}, \qquad F = \begin{pmatrix} f_0 \\ f_1 \\ f_\star \end{pmatrix} \in \mathbb{R}^3.$$

Each condition: $\mathrm{Tr}(G\, M_{ij}) + F^\top v_{ij} \geq 0$.

$$\tau^* = \sup_{\substack{0 \preccurlyeq G,\ F \in \mathbb{R}^3 \\ \mathrm{Tr}(G\, M_{\mathrm{init}}) = 1 \\ \mathrm{Tr}(G\, C_{\mathrm{alg}}) = 0 \\ \mathrm{Tr}(G\, M_{ij}) + F^\top v_{ij} \geq 0}} F^\top v_{\mathrm{perf}}$$

(i,j)	Condition
$(\star, 0)$	$f_\star \geq f_0 + \langle g_0, x_\star - x_0\rangle + \frac{\|g_0\|^2}{2L}$
$(0, \star)$	$f_0 \geq f_\star + \frac{\|g_0\|^2}{2L}$
$(\star, 1)$	$f_\star \geq f_1 + \langle g_1, x_\star - x_1\rangle + \frac{\|g_1\|^2}{2L}$
$(1, \star)$	$f_1 \geq f_\star + \frac{\|g_1\|^2}{2L}$
$(0, 1)$	$f_0 \geq f_1 + \langle g_1, x_0 - x_1\rangle + \frac{\|g_0 - g_1\|^2}{2L}$
$(1, 0)$	$f_1 \geq f_0 + \langle g_0, x_1 - x_0\rangle + \frac{\|g_1 - g_0\|^2}{2L}$

$(0,1)$: $\mathrm{Tr}(G\, M_{01}) + F^\top v_{01} \geq 0$

$$M_{01} = \begin{pmatrix} 0 & 0 & 0 & -\frac{1}{2} \\ 0 & 0 & 0 & +\frac{1}{2} \\ 0 & 0 & -\frac{1}{2L} & +\frac{1}{2L} \\ -\frac{1}{2} & +\frac{1}{2} & +\frac{1}{2L} & -\frac{1}{2L} \end{pmatrix}$$

$$v_{01} = (+1, -1, 0)^\top$$

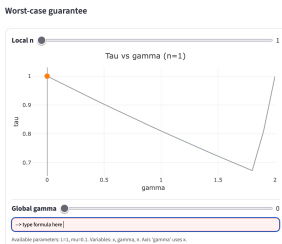
Interactive Session #2

Exploring PEP numerically with the Streamlit UI

Activity: GD on L-smooth convex functions — what is the worst-case rate for $f(x_1) - f_\star$?

1. Go to `pepitui-test.streamlit.app`
2. Select **Gradient Descent** (preimplemented) and function class $\mathcal{F}_{0,L}$ (smooth, $\mu = 0$)
3. Set $L = 1$, vary the step size γ
4. Performance metric: $f(x_1) - f_\star$ versus $\|x_0 - x_\star\|^2$
5. Run `plot` and observe the worst-case value τ^*

$\Rightarrow$ **Try to guess the formula for** τ^* as a function of L and γ.



Outline

The Generic PEP Framework

Example	$\forall f \in \mathcal{F}_{\mu,L}$	for $x_1 = x_0 - L^{-1}\nabla f(x_0)$	then	$\|x_1 - x_\star\|^2$	$\leq \tau \|x_0 - x_\star\|^2$
Generically	$\forall f \in \mathcal{F}$	for $x_T = \mathcal{A}(x_0, (\nabla f(x_t))_{t<T})$	then	$\mathrm{Perf}(x_T)$	$\leq \tau\, \mathrm{Init}(x_0)$
$\rightarrow$	Functional class	Algorithm		Worst-case guarantee	

The Generic PEP Framework

Example	$\forall f \in \mathcal{F}_{\mu,L}$	for $x_1 = x_0 - L^{-1}\nabla f(x_0)$	then	$\|x_1 - x_\star\|^2$	$\leq \tau \|x_0 - x_\star\|^2$
Generically	$\forall f \in \mathcal{F}$	for $x_T = \mathcal{A}(x_0, (\nabla f(x_t))_{t<T})$	then	$\mathrm{Perf}(x_T)$	$\leq \tau\, \mathrm{Init}(x_0)$
$\rightarrow$	Functional class	Algorithm		Worst-case guarantee	

Equivalently:

$$\tau^* = \max_{f,\, x_0,\, x_T} \frac{\mathrm{Perf}(x_T)}{\mathrm{Init}(x_0)}$$

s.t. $f \in \mathcal{F}$, $x_T = \mathcal{A}(x_0, (\nabla f(x_t))_{t<T})$

$\rightarrow$ Non-convex problem

$\rightarrow$ Infinite-dimensional over f

The Generic PEP Framework

Example	$\forall f \in \mathcal{F}_{\mu,L}$	for $x_1 = x_0 - L^{-1}\nabla f(x_0)$	then	$\|x_1 - x_\star\|^2$	$\leq \tau \|x_0 - x_\star\|^2$
Generically	$\forall f \in \mathcal{F}$	for $x_T = \mathcal{A}(x_0, (\nabla f(x_t))_{t<T})$	then	$\mathrm{Perf}(x_T)$	$\leq \tau\, \mathrm{Init}(x_0)$
$\rightarrow$	Functional class	Algorithm		Worst-case guarantee	

Equivalently:

$$\tau^* = \max_{f,\, x_0,\, x_T} \frac{\mathrm{Perf}(x_T)}{\mathrm{Init}(x_0)}$$

s.t. $f \in \mathcal{F}$, $x_T = \mathcal{A}(x_0, (\nabla f(x_t))_{t<T})$

$\rightsquigarrow$ **Sampling, Interpolation, SDP lifting**

$$\tau^* = \sup_{0 \preccurlyeq G,\ F \in \mathbb{R}^p} \mathrm{Tr}(G\, M_{\mathrm{perf}}) + F^\top v_{\mathrm{perf}}$$

$$\mathrm{Tr}(G\, M_{\mathrm{init}}) + F^\top v_{init} = 1$$

$$\forall k,\ \mathrm{Tr}(G\, M_k) + F^\top v_k = 0$$

$\rightarrow$ Non-convex problem

$\rightarrow$ Infinite-dimensional over f

The Generic PEP Framework

Example	$\forall f \in \mathcal{F}_{\mu,L}$	for $x_1 = x_0 - L^{-1}\nabla f(x_0)$	then	$\|x_1 - x_\star\|^2$	$\leq \tau \|x_0 - x_\star\|^2$
Generically	$\forall f \in \mathcal{F}$	for $x_T = \mathcal{A}(x_0, (\nabla f(x_t))_{t<T})$	then	$\mathrm{Perf}(x_T)$	$\leq \tau\, \mathrm{Init}(x_0)$
$\rightarrow$	Functional class	Algorithm		Worst-case guarantee	

Equivalently:

$$\tau^* = \max_{f,\, x_0,\, x_T} \frac{\mathrm{Perf}(x_T)}{\mathrm{Init}(x_0)}$$

s.t. $f \in \mathcal{F}$, $x_T = \mathcal{A}(x_0, (\nabla f(x_t))_{t<T})$

$\rightarrow$ Non-convex problem

$\rightarrow$ Infinite-dimensional over f

$\rightsquigarrow$ **Sampling, Interpolation, SDP lifting**

$$\tau^* = \sup_{0 \preccurlyeq G,\, F \in \mathbb{R}^p} \mathrm{Tr}(G\, M_{\mathrm{perf}}) + F^\top v_{\mathrm{perf}}$$

$$\mathrm{Tr}(G\, M_{\mathrm{init}}) + F^\top v_{init} = 1$$

$$\forall k,\ \mathrm{Tr}(G\, M_k) + F^\top v_k = 0$$

$\forall k$ aggregates algorithm & class (interpolation) constraints:

$$\begin{cases} \forall i,\ \mathrm{Tr}(G\, M_i^{\mathrm{alg}}) = 0 \\ \forall j,\ \mathrm{Tr}(G\, M_j^{\mathrm{class}}) + F^\top v_k \geq 0 \end{cases}$$

The Generic PEP Framework

Example	$\forall f \in \mathcal{F}_{\mu,L}$	for $x_1 = x_0 - L^{-1}\nabla f(x_0)$	then	$\|x_1 - x_\star\|^2$	$\leq \tau \|x_0 - x_\star\|^2$
Generically	$\forall f \in \mathcal{F}$	for $x_T = \mathcal{A}(x_0, (\nabla f(x_t))_{t<T})$	then	$\mathrm{Perf}(x_T)$	$\leq \tau\, \mathrm{Init}(x_0)$
$\rightarrow$	Functional class	Algorithm		Worst-case guarantee	

Equivalently:

$$\tau^* = \max_{f,\, x_0,\, x_T} \frac{\mathrm{Perf}(x_T)}{\mathrm{Init}(x_0)}$$

s.t. $f \in \mathcal{F}$, $x_T = \mathcal{A}(x_0, (\nabla f(x_t))_{t<T})$

$\rightarrow$ Non-convex problem

$\rightarrow$ Infinite-dimensional over f

$\rightsquigarrow$ **Sampling, Interpolation, SDP lifting**

$$\tau^* = \sup_{0 \preccurlyeq G,\, F \in \mathbb{R}^p} \mathrm{Tr}(G\, M_{\mathrm{perf}}) + F^\top v_{\mathrm{perf}}$$
$$\mathrm{Tr}(G\, M_{\mathrm{init}}) + F^\top v_{init} = 1$$
$$\forall k,\ \mathrm{Tr}(G\, M_k) + F^\top v_k = 0$$

$\forall k$ aggregates algorithm & class (interpolation) constraints:

$$\begin{cases} \forall i,\ \mathrm{Tr}(G\, M_i^{\mathrm{alg}}) = 0 \\ \forall j,\ \mathrm{Tr}(G\, M_j^{\mathrm{class}}) + F^\top v_k \geq 0 \end{cases}$$

Worst-case guarantees with PEP

$\rightarrow$ Finding a tight WCG = solving a convex SDP

$\heartsuit$ Automatic numerical WCG $\rightarrow$ design & proof-checking

$\heartsuit$ Works for many algorithms & function classes

PEPit: computer-assisted worst-case analysis in Python

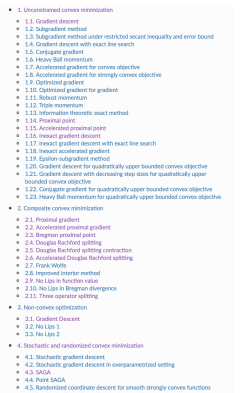
Goals of PEPit:

- Avoids manual SDP modeling steps
- Collaborative, easy-to-use methodology
- Code as close as possible to mathematical specification

Supported setups:

- Algorithms: Gradient, Prox, Inexact-Gradient, Line-search, ...
- Function classes: smooth, strongly convex, quadratically upper bounded, ...
- Performance metrics: $f(x_T) - f_\star$, $\|x_T - x_\star\|^2$, $\|\nabla f(x_T)\|^2$, ...

- **80+ documented examples**



- Complete docs, active community

What can we do with PEP?

Performance metrics: any linear combination of $f(x_T) - f_\star$, $\|x_T - x_\star\|^2$, $\|\nabla f(x_T)\|^2$ at various timesteps.

What can we do with PEP?

Performance metrics: any linear combination of $f(x_T) - f_\star$, $\|x_T - x_\star\|^2$, $\|\nabla f(x_T)\|^2$ at various timesteps.

Many function classes:

- L-smooth, μ-strongly convex
- Convex + quadratically upper bounded
- Non-smooth (Lipschitz), non-convex
- Indicator functions of convex bounded sets, balls
- Functions with bounded gradients
- ...

What can we do with PEP?

Performance metrics: any linear combination of $f(x_T) - f_\star$, $\|x_T - x_\star\|^2$, $\|\nabla f(x_T)\|^2$ at various timesteps.

Many function classes:

- L-smooth, μ-strongly convex
- Convex + quadratically upper bounded
- Non-smooth (Lipschitz), non-convex
- Indicator functions of convex bounded sets, balls
- Functions with bounded gradients
- ...

Many algorithms:

- GD, heavy ball, Nesterov's NAG
- Proximal point, ADMM, Douglas-Rachford
- Inexact updates
- Line Search
- Stochastic methods (SGD, variance reduction)
- ...

What can we do with PEP?

Performance metrics: any linear combination of $f(x_T) - f_\star$, $\|x_T - x_\star\|^2$, $\|\nabla f(x_T)\|^2$ at various timesteps.

Many function classes:

- L-smooth, μ-strongly convex
- Convex + quadratically upper bounded
- Non-smooth (Lipschitz), non-convex
- Indicator functions of convex bounded sets, balls
- Functions with bounded gradients
- ...

Many algorithms:

- GD, heavy ball, Nesterov's NAG
- Proximal point, ADMM, Douglas-Rachford
- Inexact updates
- Line Search
- Stochastic methods (SGD, variance reduction)
- ...

For *all* of these, the worst-case rate is the solution of an SDP.

PEPit solves it in a few lines of Python.

EXAMPLES

—

Beyond gradient descent: more oracles in PEP

Example 1: Proximal Step to Minimize $f \in \mathcal{F}_{\mu,L}$

Algorithm. $x_{k+1} = \mathrm{prox}_{\gamma f}(x_k) := \arg\min_x \left\{ f(x) + \frac{1}{2\gamma}\|x - x_k\|^2 \right\}$.

Example 1: Proximal Step to Minimize $f \in \mathcal{F}_{\mu,L}$

Algorithm. $x_{k+1} = \mathrm{prox}_{\gamma f}(x_k) := \arg\min_x \left\{ f(x) + \frac{1}{2\gamma}\|x - x_k\|^2 \right\}$.

Equivalently, the implicit gradient step:

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

Example 1: Proximal Step to Minimize $f \in \mathcal{F}_{\mu,L}$

Algorithm. $x_{k+1} = \mathrm{prox}_{\gamma f}(x_k) := \arg\min_x \left\{ f(x) + \frac{1}{2\gamma}\|x - x_k\|^2 \right\}$.

Equivalently, the implicit gradient step:

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

In PEP. Sample two triplets:

- $(x_k,\ g_k = \nabla f(x_k),\ f_k = f(x_k))$
- $(x_{k+1},\ g_{k+1} = \nabla f(x_{k+1}),\ f_{k+1} = f(x_{k+1}))$

Algorithm constraint: $x_{k+1} = x_k - \gamma\, g_{k+1}$

i.e. $\mathrm{Tr}(G\, C_{\mathrm{alg}}) = 0$ (linear in G).

Class constraint:

Interpolation conditions for $f \in \mathcal{F}_{\mu,L}$ on triplets $(x_k, g_k, g(x_k))$ and $(x_{k+1}, g_{k+1}, g(x_{k+1}))$.

Example 1: Proximal Step to Minimize $f \in \mathcal{F}_{\mu,L}$

Algorithm. $x_{k+1} = \mathrm{prox}_{\gamma f}(x_k) := \arg\min_x \left\{ f(x) + \frac{1}{2\gamma}\|x - x_k\|^2 \right\}$.

Equivalently, the implicit gradient step:

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

In PEP. Sample two triplets:

- $(x_k,\ g_k = \nabla f(x_k),\ f_k = f(x_k))$
- $(x_{k+1},\ g_{k+1} = \nabla f(x_{k+1}),\ f_{k+1} = f(x_{k+1}))$

Algorithm constraint: $x_{k+1} = x_k - \gamma\, g_{k+1}$

i.e. $\mathrm{Tr}(G\, C_{\mathrm{alg}}) = 0$ (linear in G).

Class constraint:

Interpolation conditions for $f \in \mathcal{F}_{\mu,L}$ on triplets $(x_k, g_k, g(x_k))$ and $(x_{k+1}, g_{k+1}, g(x_{k+1}))$.

PEPit.

- `f = problem.declare_function( SmoothStronglyConvexFunction, mu=mu, L=L)`
- `x1, g1, f1 = proximal_step(x0, g, gamma)`

Example 1: Proximal Step to Minimize $f \in \mathcal{F}_{\mu,L}$

Algorithm. $x_{k+1} = \mathrm{prox}_{\gamma f}(x_k) := \arg\min_x \left\{ f(x) + \frac{1}{2\gamma}\|x - x_k\|^2 \right\}$.

Equivalently, the implicit gradient step:

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

In PEP. Sample two triplets:

- $(x_k,\ g_k = \nabla f(x_k),\ f_k = f(x_k))$
- $(x_{k+1},\ g_{k+1} = \nabla f(x_{k+1}),\ f_{k+1} = f(x_{k+1}))$

Algorithm constraint: $x_{k+1} = x_k - \gamma\, g_{k+1}$

i.e. $\mathrm{Tr}(G\, C_{\mathrm{alg}}) = 0$ (linear in G).

Class constraint:

Interpolation conditions for $f \in \mathcal{F}_{\mu,L}$ on triplets $(x_k, g_k, g(x_k))$ and $(x_{k+1}, g_{k+1}, g(x_{k+1}))$.

PEPit.

- `f = problem.declare_function( SmoothStronglyConvexFunction, mu=mu, L=L)`
- `x1, g1, f1 = proximal_step(x0, g, gamma)`

💡 - The implicit equation

$$x_{k+1} = x_k - \gamma \nabla f(x_{k+1})$$

is *not solved explicitly*: it is encoded as an algorithm constraint (+ interpolation conditions)

No closed form for $\mathrm{prox}_{\gamma f}$ is needed.

Example 2: Projection onto a Convex Set

Algorithm. Projected gradient descent:

$$x_{k+1} = \Pi_C(x_k - \gamma \nabla f(x_k)), \qquad \Pi_C(y) = \operatorname*{arg\,min}_{x \in C} \|x - y\|.$$

Example 2: Projection onto a Convex Set

Algorithm. Projected gradient descent:

$$x_{k+1} = \Pi_C(x_k - \gamma \nabla f(x_k)), \qquad \Pi_C(y) = \operatorname*{arg\,min}_{x \in C} \|x - y\|.$$

Connection to PEP. The indicator function $\iota_C(x) = 0$ if $x \in C$, $+\infty$ otherwise, satisfies:

$$\Pi_C(y) = \operatorname{prox}_{\iota_C}(y).$$

ι_C is a convex function $\rightarrow$ its proximal operator is covered by the **same interpolation conditions** as any convex g.

Example 2: Projection onto a Convex Set

Algorithm. Projected gradient descent:

$$x_{k+1} = \Pi_C(x_k - \gamma \nabla f(x_k)), \qquad \Pi_C(y) = \operatorname*{arg\,min}_{x \in C} \|x - y\|.$$

Connection to PEP. The indicator function $\iota_C(x) = 0$ if $x \in C$, $+\infty$ otherwise, satisfies:

$$\Pi_C(y) = \operatorname{prox}_{\iota_C}(y).$$

ι_C is a convex function $\rightarrow$ its proximal operator is covered by the **same interpolation conditions** as any convex g.

💡 **Indicator functions of bounded sets, and of balls are interpolable**

$\rightarrow$ see Convex indicator, Pepit doc.

Example 3: Inexact Gradient Step

Algorithm. Instead of the exact gradient, use an inexact oracle $\tilde{g}_k$ satisfying:

$$\|\tilde{g}_k - \nabla f(x_k)\| \leq \delta \quad \text{or} \quad \|\tilde{g}_k - \nabla f(x_k)\|^2 \leq \delta^2 \|\nabla f(x_k)\|^2.$$

Step: $x_{k+1} = x_k - \gamma\, \tilde{g}_k$.

Example 3: Inexact Gradient Step

Algorithm. Instead of the exact gradient, use an inexact oracle $\tilde{g}_k$ satisfying:

$$\|\tilde{g}_k - \nabla f(x_k)\| \leq \delta \quad \text{or} \quad \|\tilde{g}_k - \nabla f(x_k)\|^2 \leq \delta^2 \|\nabla f(x_k)\|^2.$$

Step: $x_{k+1} = x_k - \gamma\, \tilde{g}_k$.

In PEP:

- Sample exact gradient $g_k = \nabla f(x_k)$ (interpolation conditions for f).
- Introduce $\tilde{g}_k$ as an **additional variable**.
- The inexactness condition is a **quadratic constraint** in G:

$$\|\tilde{g}_k - g_k\|^2 \leq \delta^2 \;\Leftrightarrow\; \mathrm{Tr}(G\, M_{\text{inexact}}) \leq \delta^2.$$

Example 3: Inexact Gradient Step

Algorithm. Instead of the exact gradient, use an inexact oracle $\tilde{g}_k$ satisfying:

$$\|\tilde{g}_k - \nabla f(x_k)\| \leq \delta \quad \text{or} \quad \|\tilde{g}_k - \nabla f(x_k)\|^2 \leq \delta^2 \|\nabla f(x_k)\|^2.$$

Step: $x_{k+1} = x_k - \gamma\, \tilde{g}_k$.

In PEP:

- Sample exact gradient $g_k = \nabla f(x_k)$ (interpolation conditions for f).
- Introduce $\tilde{g}_k$ as an **additional variable**.
- The inexactness condition is a **quadratic constraint** in G:

$$\|\tilde{g}_k - g_k\|^2 \leq \delta^2 \;\Leftrightarrow\; \mathrm{Tr}(G\, M_{\text{inexact}}) \leq \delta^2.$$

💡 inexactness = one extra quadratic constraint.

PEP & PEPit handle this via `problem.add_constraint(...)`.

→ see Inexact gradient step oracle, Pepit doc.

Example 4: Line Search

Algorithm. Exact line search along the gradient direction:

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

Example 4: Line Search

Algorithm. Exact line search along the gradient direction:

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

True oracle condition. x_{k+1} is the exact minimizer of $\phi(\gamma) = f(x_k - \gamma g_k)$, so it must satisfy:

$$f(x_{k+1}) \;\leq\; f(x_k - \gamma \, g_k) \quad \forall \, \gamma \geq 0.$$

This is an **infinite** family of constraints: one for every γ.

Example 4: Line Search

Algorithm. Exact line search along the gradient direction:

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

True oracle condition. x_{k+1} is the exact minimizer of $\phi(\gamma) = f(x_k - \gamma g_k)$, so it must satisfy:

$$f(x_{k+1}) \;\leq\; f(x_k - \gamma \, g_k) \quad \forall \, \gamma \geq 0.$$

This is an **infinite** family of constraints: one for every γ.

Relaxation used in PEP.

1. First-order optimality of ϕ at γ_k:

$$\phi'(\gamma_k) = 0 \;\Leftrightarrow\; \langle \nabla f(x_{k+1}), \, \nabla f(x_k) \rangle = 0.$$

Example 4: Line Search

Algorithm. Exact line search along the gradient direction:

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

True oracle condition. x_{k+1} is the exact minimizer of $\phi(\gamma) = f(x_k - \gamma g_k)$, so it must satisfy:

$$f(x_{k+1}) \;\leq\; f(x_k - \gamma \, g_k) \quad \forall \, \gamma \geq 0.$$

This is an **infinite** family of constraints: one for every γ.

Relaxation used in PEP.

1. First-order optimality of ϕ at γ_k:

$$\phi'(\gamma_k) = 0 \;\Leftrightarrow\; \langle \nabla f(x_{k+1}), \, \nabla f(x_k) \rangle = 0.$$

2. $x_{k+1} = x_k - \gamma_k \, \nabla f(x_k) \Rightarrow \langle x_{k+1} - x_k, \nabla f(x_{k+1}) \rangle = 0$

Example 4: Line Search

Algorithm. Exact line search along the gradient direction:

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

True oracle condition. x_{k+1} is the exact minimizer of $\phi(\gamma) = f(x_k - \gamma g_k)$, so it must satisfy:

$$f(x_{k+1}) \;\leq\; f(x_k - \gamma \, g_k) \quad \forall \, \gamma \geq 0.$$

This is an **infinite** family of constraints: one for every γ.

Relaxation used in PEP.

1. First-order optimality of ϕ at γ_k:

$$\phi'(\gamma_k) = 0 \;\Leftrightarrow\; \langle \nabla f(x_{k+1}), \, \nabla f(x_k) \rangle = 0.$$

2. $x_{k+1} = x_k - \gamma_k \, \nabla f(x_k) \Rightarrow \langle x_{k+1} - x_k, \nabla f(x_{k+1}) \rangle = 0$

These are two **linear constraints** in G:

$$\mathrm{Tr}(G \, M_{\mathrm{ls}}) = 0.$$

Example 4: Line Search

Algorithm. Exact line search along the gradient direction:

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

True oracle condition. x_{k+1} is the exact minimizer of $\phi(\gamma) = f(x_k - \gamma g_k)$, so it must satisfy:

$$f(x_{k+1}) \;\leq\; f(x_k - \gamma\, g_k) \quad \forall\, \gamma \geq 0.$$

This is an **infinite** family of constraints: one for every γ.

Relaxation used in PEP.

1. First-order optimality of ϕ at γ_k:

$$\phi'(\gamma_k) = 0 \;\Leftrightarrow\; \langle \nabla f(x_{k+1}),\, \nabla f(x_k) \rangle = 0.$$

2. $x_{k+1} = x_k - \gamma_k \nabla f(x_k) \Rightarrow \langle x_{k+1} - x_k, \nabla f(x_{k+1}) \rangle = 0$

These are two **linear constraints** in G:

$$\mathrm{Tr}(G\, M_{\mathrm{ls}}) = 0.$$

It's a relaxation!

Orthogonality is *necessary* for exact line search.

The feasible set of PEP is **strictly larger** than the set of true line-search instances.

$\rightarrow$ Rates are **valid** but not necessarily **tight**.

Other examples

See the documentation.

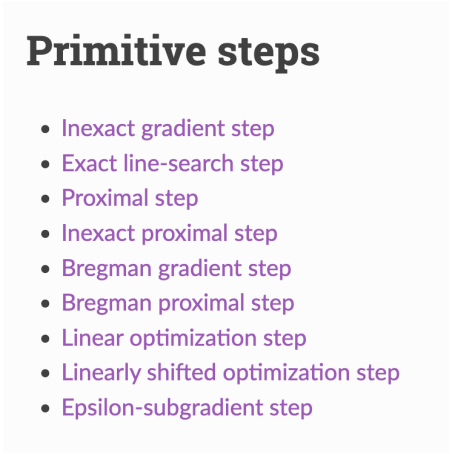


Figure: Primitive steps in Pepit

Outline

Minimal PEPit Example: GD on smooth convex f – MWE

Goal: find the tightest τ such that $f(x_n) - f_\star \leq \tau \|x_0 - x_\star\|^2$ for GD with step $\gamma = 1/L$.

Minimal PEPit Example: GD on smooth convex f – MWE

Goal: find the tightest τ such that $f(x_n) - f_\star \leq \tau \|x_0 - x_\star\|^2$ for GD with step $\gamma = 1/L$.

```python
from PEPit import PEP
from PEPit.functions import SmoothConvexFunction

# --- Problem parameters ---
L, gamma, n = 1.0, 1.0, 3  # smoothness, step size, nb of steps

# --- Set up the PEP ---
problem = PEP()

# Declare the function class
func = problem.declare_function(SmoothConvexFunction, L=L)

# Define the optimum x_star and f_star
x_star = func.stationary_point()
f_star = func(x_star)

# Define the starting point
x0 = problem.set_initial_point()

# Initial condition: ||x0 - x_star||^2 <= 1
problem.set_initial_condition((x0 - x_star)**2 <= 1)

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

# Set performance metric: f(x_n) - f_star
problem.set_performance_metric(func(x) - f_star)

# Solve the SDP
tau = problem.solve(verbose=1)
print(f"Worst-case rate: {tau:.6f}")  # should give L/(4n+2) =
    1/6 for n=1
```

Figure: Pepit Output

Minimal PEPit Example: GD on smooth convex f – MWE

Goal: find the tightest τ such that $f(x_n) - f_\star \leq \tau \|x_0 - x_\star\|^2$ for GD with step $\gamma = 1/L$.

```python
from PEPit import PEP
from PEPit.functions import SmoothConvexFunction

# --- Problem parameters ---
L, gamma, n = 1.0, 1.0, 3  # smoothness, step size, nb of steps

# --- Set up the PEP ---
problem = PEP()

# Declare the function class
func = problem.declare_function(SmoothConvexFunction, L=L)

# Define the optimum x_star and f_star
x_star = func.stationary_point()
f_star = func(x_star)

# Define the starting point
x0 = problem.set_initial_point()

# Initial condition: ||x0 - x_star||^2 <= 1
problem.set_initial_condition((x0 - x_star)**2 <= 1)

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

# Set performance metric: f(x_n) - f_star
problem.set_performance_metric(func(x) - f_star)

# Solve the SDP
tau = problem.solve(verbose=1)
print(f"Worst-case rate: {tau:.6f}")  # should give L/(4n+2) =
    1/6 for n=1
```

```
(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 7 scalar constraint(s) ...
                        Function 1 : 7 scalar constraint(s) added
(PEPit) Setting up the problem: additional constraints for 0 function(s)
(PEPit) Setting up the problem: size of the Gram matrix: 7
(PEPit) Compiling SDP
(PEPit) Calling SDP solver
(PEPit) Solver status: optimal (wrapper:cvxpy, solver: SCS); optimal value: 0.0714275603704179
(PEPit) Primal feasibility check:
                The solver found a Gram matrix that is positive semi-definite
                All the primal scalar constraints are verified up to an error of 8.748184794626912e-08
(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
(PEPit) The worst-case guarantee proof is perfectly reconstituted up to an error of 3.4133737357895655e-06
(PEPit) Final upper bound (dual): 0.07142813009889529 and lower bound (primal example): 0.0714275603704179
(PEPit) Duality gap: absolute: 5.697284773814593e-07 and relative: 7.976311586548534e-06
Worst-case rate: 0.071428
```

Figure: Pepit Output

Minimal PEPit Example: GD on smooth convex f – MWE

Goal: find the tightest τ such that $f(x_n) - f_\star \leq \tau \|x_0 - x_\star\|^2$ for GD with step $\gamma = 1/L$.

```python
from PEPit import PEP
from PEPit.functions import SmoothConvexFunction

# --- Problem parameters ---
L, gamma, n = 1.0, 1.0, 3  # smoothness, step size, nb of steps

# --- Set up the PEP ---
problem = PEP()

# Declare the function class
func = problem.declare_function(SmoothConvexFunction, L=L)

# Define the optimum x_star and f_star
x_star = func.stationary_point()
f_star = func(x_star)

# Define the starting point
x0 = problem.set_initial_point()

# Initial condition: ||x0 - x_star||^2 <= 1
problem.set_initial_condition((x0 - x_star)**2 <= 1)

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

# Set performance metric: f(x_n) - f_star
problem.set_performance_metric(func(x) - f_star)

# Solve the SDP
tau = problem.solve(verbose=1)
print(f"Worst-case rate: {tau:.6f}")  # should give L/(4n+2) =
    1/6 for n=1
```

```
(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 20 scalar constraint(s) ...
                        Function 1 : 20 scalar constraint(s) added
(PEPit) Setting up the problem: additional constraints for 0 function(s)
(PEPit) Setting up the problem: size of the Gram matrix: 6x6
(PEPit) Compiling SDP
(PEPit) Calling SDP solver
(PEPit) Solver status: optimal (wrapper:cvxpy, solver: SCS); optimal value: 0.0714275603704179
(PEPit) Primal feasibility check:
                The solver found a Gram matrix that is positive semi-definite
                All the primal scalar constraints are verified up to an error of 8.748184794626912e-08
(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
(PEPit) The worst-case guarantee proof is perfectly reconstituted up to an error of 3.4133737357895655e-06
(PEPit) Final upper bound (dual): 0.07142813009889529 and lower bound (primal example): 0.0714275603704179
(PEPit) Duality gap: absolute: 5.697284773814593e-07 and relative: 7.976311586548534e-06
Worst-case rate: 0.071428
```

Figure: Pepit Output

PEPit: what happens under the hood?

What you write (math):

$$\tau^* = \max_{f,\, x_0,\, x_T} \frac{\mathrm{Perf}(x_T)}{\mathrm{Init}(x_0)}$$

s.t. $f \in \mathcal{F}$, $x_T = \mathcal{A}(x_0, (\nabla f(x_t))_{t<T})$

What PEPit solves (SDP):

$$\tau^* = \sup_{0 \preccurlyeq G,\, F \in \mathbb{R}^p} \mathrm{Tr}(G\, M_{\mathrm{perf}}) + F^\top v_{\mathrm{perf}}$$

$$\mathrm{Tr}(G\, M_{\mathrm{init}}) + F^\top v_{init} = 1$$

$$\forall k,\ \mathrm{Tr}(G\, M_k) + F^\top v_k = 0$$

PEPit: what happens under the hood?

What you write (math):

$$\tau^* = \max_{f,\, x_0,\, x_T} \frac{\mathrm{Perf}(x_T)}{\mathrm{Init}(x_0)}$$

s.t. $f \in \mathcal{F}$, $x_T = \mathcal{A}(x_0, (\nabla f(x_t))_{t<T})$

What PEPit solves (SDP):

$$\tau^* = \sup_{0 \preccurlyeq G,\, F \in \mathbb{R}^p} \mathrm{Tr}(G\, M_{\mathrm{perf}}) + F^\top v_{\mathrm{perf}}$$

$$\mathrm{Tr}(G\, M_{\mathrm{init}}) + F^\top v_{init} = 1$$

$$\forall k,\ \mathrm{Tr}(G\, M_k) + F^\top v_k = 0$$

PEPit automatically:

1. Builds the Gram matrix G from the declared points.
2. Encodes algorithm constraints as linear equalities in G, as well as other constraints.
3. Encodes interpolation conditions as linear inequalities in G.
4. Calls an SDP solver (e.g. MOSEK, SCS, CVXPY).
5. Returns the optimal rate + primal certificate (worst-case function).

PEPit: what happens under the hood?

What you write (math):

$$\tau^* = \max_{f,\, x_0,\, x_T} \frac{\mathrm{Perf}(x_T)}{\mathrm{Init}(x_0)}$$

s.t. $f \in \mathcal{F}$, $x_T = \mathcal{A}(x_0, (\nabla f(x_t))_{t<T})$

What PEPit solves (SDP):

$$\tau^* = \sup_{0 \preccurlyeq G,\, F \in \mathbb{R}^p} \mathrm{Tr}(G\, M_{\mathrm{perf}}) + F^\top v_{\mathrm{perf}}$$

$$\mathrm{Tr}(G\, M_{\mathrm{init}}) + F^\top v_{init} = 1$$

$$\forall k,\ \mathrm{Tr}(G\, M_k) + F^\top v_k = 0$$

PEPit automatically:

1. Builds the Gram matrix G from the declared points.
2. Encodes algorithm constraints as linear equalities in G, as well as other constraints.
3. Encodes interpolation conditions as linear inequalities in G.
4. Calls an SDP solver (e.g. MOSEK, SCS, CVXPY).
5. Returns the optimal rate + primal certificate (worst-case function).

Output for $n = 1$, $L = 1$, $\gamma = 1/L$

- Worst-case rate: $\tau^* \approx 0.1667$

Outline

What does the primal SDP tell us?

The PEP primal SDP (generic form):

$$\tau^* = \sup_{0 \preccurlyeq G,\, F \in \mathbb{R}^p} \quad \mathrm{Tr}(G\, M_{\text{perf}}) + F^\top v_{\text{perf}}$$
$$\mathrm{Tr}(G\, M_{\text{init}}) + F^\top v_{init} = 1$$
$$\forall k,\ \mathrm{Tr}(G\, M_k) + F^\top v_k = 0$$

with the list corresponding to the list of algorithmic and class constraints: $\mathrm{Tr}(G\, C_{\ell,\text{alg}}) = 0, \mathrm{Tr}(G\, M_{ij}) + F^\top v_{ij} \geq 0,\ \forall i, j$

What does the primal SDP tell us?

The PEP primal SDP (generic form):

$$\tau^* = \sup_{0 \preccurlyeq G,\ F \in \mathbb{R}^p} \ \mathrm{Tr}(G\, M_{\text{perf}}) + F^\top v_{\text{perf}}$$
$$\mathrm{Tr}(G\, M_{\text{init}}) + F^\top v_{init} = 1$$
$$\forall k,\ \mathrm{Tr}(G\, M_k) + F^\top v_k = 0$$

with the list corresponding to the list of algorithmic and class constraints: $\mathrm{Tr}(G\, C_{\ell,\text{alg}}) = 0, \mathrm{Tr}(G\, M_{ij}) + F^\top v_{ij} \geq 0,\ \forall i, j$

G is a **Gram matrix**:

$$G = \begin{pmatrix} x_0 \\ \vdots \\ g_k \end{pmatrix} \begin{pmatrix} x_0 \\ \vdots \\ g_k \end{pmatrix}^\top \succcurlyeq 0$$

Its **entries** are inner products of iterates and gradients.

What does the primal SDP tell us?

The PEP primal SDP (generic form):

$$\tau^* = \sup_{0 \preccurlyeq G,\, F \in \mathbb{R}^p} \quad \mathrm{Tr}(G\, M_{\text{perf}}) + F^\top v_{\text{perf}}$$
$$\mathrm{Tr}(G\, M_{\text{init}}) + F^\top v_{init} = 1$$
$$\forall k,\ \mathrm{Tr}(G\, M_k) + F^\top v_k = 0$$

with the list corresponding to the list of algorithmic and class constraints: $\mathrm{Tr}(G\, C_{\ell,\text{alg}}) = 0, \mathrm{Tr}(G\, M_{ij}) + F^\top v_{ij} \geq 0,\ \forall i, j$

G is a **Gram matrix**:

$$G = \begin{pmatrix} x_0 \\ \vdots \\ g_k \end{pmatrix} \begin{pmatrix} x_0 \\ \vdots \\ g_k \end{pmatrix}^\top \succcurlyeq 0$$

Its **entries** are inner products of iterates and gradients.

Key observation: primal feasible ⇒ lower bound

What does the primal SDP tell us?

The PEP primal SDP (generic form):

$$\tau^* = \sup_{0 \preccurlyeq G,\ F \in \mathbb{R}^p} \ \mathrm{Tr}(G\, M_{\text{perf}}) + F^\top v_{\text{perf}}$$
$$\mathrm{Tr}(G\, M_{\text{init}}) + F^\top v_{init} = 1$$
$$\forall k,\ \mathrm{Tr}(G\, M_k) + F^\top v_k = 0$$

with the list corresponding to the list of algorithmic and class constraints: $\mathrm{Tr}(G\, C_{\ell,\text{alg}}) = 0, \mathrm{Tr}(G\, M_{ij}) + F^\top v_{ij} \geq 0,\ \forall i, j$

G is a **Gram matrix**:

$$G = \begin{pmatrix} x_0 \\ \vdots \\ g_k \end{pmatrix} \begin{pmatrix} x_0 \\ \vdots \\ g_k \end{pmatrix}^\top \succcurlyeq 0$$

Its **entries** are inner products of iterates and gradients.

Key observation: primal feasible ⇒ lower bound

Any feasible $G \succcurlyeq 0$ and F (i.e., satisfying the constraints) gives a **valid** instance:

What does the primal SDP tell us?

The PEP primal SDP (generic form):

$$\tau^* = \sup_{0 \preccurlyeq G,\, F \in \mathbb{R}^p} \ \mathrm{Tr}(G\, M_{\text{perf}}) + F^\top v_{\text{perf}}$$
$$\mathrm{Tr}(G\, M_{\text{init}}) + F^\top v_{init} = 1$$
$$\forall k,\ \mathrm{Tr}(G\, M_k) + F^\top v_k = 0$$

with the list corresponding to the list of algorithmic and class constraints: $\mathrm{Tr}(G\, C_{\ell,\text{alg}}) = 0, \mathrm{Tr}(G\, M_{ij}) + F^\top v_{ij} \geq 0,\ \forall i, j$

G is a **Gram matrix**:

$$G = \begin{pmatrix} x_0 \\ \vdots \\ g_k \end{pmatrix} \begin{pmatrix} x_0 \\ \vdots \\ g_k \end{pmatrix}^\top \succcurlyeq 0$$

Its **entries** are inner products of iterates and gradients.

Key observation: primal feasible ⇒ lower bound

Any feasible $G \succcurlyeq 0$ and F (i.e., satisfying the constraints) gives a **valid** instance:

- **any points/gradients** x_i, g_i with Gram G (e.g., recovered through Cholesky / eigendecomposition)

What does the primal SDP tell us?

The PEP primal SDP (generic form):

$$\tau^* = \sup_{0 \preccurlyeq G,\, F \in \mathbb{R}^p} \quad \mathrm{Tr}(G\, M_{\text{perf}}) + F^\top v_{\text{perf}}$$
$$\mathrm{Tr}(G\, M_{\text{init}}) + F^\top v_{init} = 1$$
$$\forall k,\ \mathrm{Tr}(G\, M_k) + F^\top v_k = 0$$

with the list corresponding to the list of algorithmic and class constraints: $\mathrm{Tr}(G\, C_{\ell,\text{alg}}) = 0, \mathrm{Tr}(G\, M_{ij}) + F^\top v_{ij} \geq 0,\ \forall i, j$

G is a **Gram matrix**:

$$G = \begin{pmatrix} x_0 \\ \vdots \\ g_k \end{pmatrix} \begin{pmatrix} x_0 \\ \vdots \\ g_k \end{pmatrix}^\top \succcurlyeq 0$$

Its **entries** are inner products of iterates and gradients.

Key observation: primal feasible ⇒ lower bound

Any feasible $G \succcurlyeq 0$ and F (i.e., satisfying the constraints) gives a **valid** instance:

- **any points/gradients** x_i, g_i with Gram G (e.g., recovered through Cholesky / eigendecomposition)
- build an interpolating function $f \in \mathcal{F}$ through these points

What does the primal SDP tell us?

The PEP primal SDP (generic form):

$$\tau^* = \sup_{0 \preccurlyeq G,\, F \in \mathbb{R}^p} \quad \mathrm{Tr}(G\, M_{\text{perf}}) + F^\top v_{\text{perf}}$$
$$\mathrm{Tr}(G\, M_{\text{init}}) + F^\top v_{init} = 1$$
$$\forall k,\ \mathrm{Tr}(G\, M_k) + F^\top v_k = 0$$

with the list corresponding to the list of algorithmic and class constraints: $\mathrm{Tr}(G\, C_{\ell,\text{alg}}) = 0, \mathrm{Tr}(G\, M_{ij}) + F^\top v_{ij} \geq 0,\ \forall i, j$

G is a **Gram matrix**:

$$G = \begin{pmatrix} x_0 \\ \vdots \\ g_k \end{pmatrix} \begin{pmatrix} x_0 \\ \vdots \\ g_k \end{pmatrix}^\top \succcurlyeq 0$$

Its **entries** are inner products of iterates and gradients.

Key observation: primal feasible ⇒ lower bound

Any feasible $G \succcurlyeq 0$ and F (i.e., satisfying the constraints) gives a **valid** instance:

- **any points/gradients** x_i, g_i with Gram G (e.g., recovered through Cholesky / eigendecomposition)
- build an interpolating function $f \in \mathcal{F}$ through these points
- For which the algorithm on this function generates the sequence.

What does the primal SDP tell us?

The PEP primal SDP (generic form):

$$\tau^* = \sup_{0 \preccurlyeq G,\, F \in \mathbb{R}^p} \ \mathrm{Tr}(G\, M_{\text{perf}}) + F^\top v_{\text{perf}}$$
$$\mathrm{Tr}(G\, M_{\text{init}}) + F^\top v_{init} = 1$$
$$\forall k,\ \mathrm{Tr}(G\, M_k) + F^\top v_k = 0$$

with the list corresponding to the list of algorithmic and class constraints: $\mathrm{Tr}(G\, C_{\ell,\text{alg}}) = 0, \mathrm{Tr}(G\, M_{ij}) + F^\top v_{ij} \geq 0,\ \forall i, j$

G is a **Gram matrix**:

$$G = \begin{pmatrix} x_0 \\ \vdots \\ g_k \end{pmatrix} \begin{pmatrix} x_0 \\ \vdots \\ g_k \end{pmatrix}^\top \succcurlyeq 0$$

Its **entries** are inner products of iterates and gradients.

Key observation: primal feasible ⇒ lower bound

Any feasible $G \succcurlyeq 0$ and F (i.e., satisfying the constraints) gives a **valid** instance:

- **any points/gradients** x_i, g_i with Gram G (e.g., recovered through Cholesky / eigendecomposition)
- build an interpolating function $f \in \mathcal{F}$ through these points
- For which the algorithm on this function generates the sequence.

⇒ explicit algorithm trajectory with $\mathrm{Perf}(x_T) = \mathrm{Tr}(G\, M_{\text{perf}})$; and $\mathrm{Init}(x_0) = \mathrm{Tr}(G\, M_{\text{init}})$

From the Gram matrix $G^\star$ to a counter-example

Example: one step of GD on $\mathcal{F}_{0,L}$, metric $f(x_1) - f_\star \leq \tau \|x_0 - x_\star\|^2$.

From the Gram matrix $G^\star$ to a counter-example

Example: one step of GD on $\mathcal{F}_{0,L}$, metric $f(x_1) - f_\star \leq \tau \|x_0 - x_\star\|^2$.

After solving the SDP, the solver returns $G^\star$ with objective value $\tau^\star$.

From the Gram matrix $G^\star$ to a counter-example

Example: one step of GD on $\mathcal{F}_{0,L}$, metric $f(x_1) - f_\star \leq \tau \|x_0 - x_\star\|^2$.

After solving the SDP, the solver returns $G^\star$ with objective value $\tau^\star$.

Step 1: extract points. Factorise $G^\star = V^\top V$ (e.g. with a Cholesky dec.).
The columns of V are (the d-dimensional representatives of) x_0, x_1, g_0, g_1.

From the Gram matrix $G^\star$ to a counter-example

Example: one step of GD on $\mathcal{F}_{0,L}$, metric $f(x_1) - f_\star \leq \tau \|x_0 - x_\star\|^2$.

After solving the SDP, the solver returns $G^\star$ with objective value $\tau^\star$.

Step 1: extract points. Factorise $G^\star = V^\top V$ (e.g. with a Cholesky dec.).
The columns of V are (the d-dimensional representatives of) x_0, x_1, g_0, g_1.

Step 2: build the worst-case function. By the interpolation theorem, there exists $f \in \mathcal{F}_{0,L}$ such that $f(x_i) = f_i$ and $\nabla f(x_i) = g_i$ for all i.
Take e.g. the *smallest such* interpolant.

From the Gram matrix $G^\star$ to a counter-example

Example: one step of GD on $\mathcal{F}_{0,L}$, metric $f(x_1) - f_\star \leq \tau \|x_0 - x_\star\|^2$.

After solving the SDP, the solver returns $G^\star$ with objective value $\tau^\star$.

Step 1: extract points. Factorise $G^\star = V^\top V$ (e.g. with a Cholesky dec.).
The columns of V are (the d-dimensional representatives of) x_0, x_1, g_0, g_1.

Step 2: build the worst-case function. By the interpolation theorem, there exists $f \in \mathcal{F}_{0,L}$ such that $f(x_i) = f_i$ and $\nabla f(x_i) = g_i$ for all i.
Take e.g. the *smallest such* interpolant.

Result: (f, x_0) is an explicit instance achieving

$$f(x_1) - f_\star = \tau^\star \|x_0 - x_\star\|^2.$$

No algorithm can do better on every function in $\mathcal{F}_{0,L}$.

From the Gram matrix $G^\star$ to a counter-example

Example: one step of GD on $\mathcal{F}_{0,L}$, metric $f(x_1) - f_\star \leq \tau \|x_0 - x_\star\|^2$.

After solving the SDP, the solver returns $G^\star$ with objective value $\tau^\star$.

Step 1: extract points. Factorise $G^\star = V^\top V$ (e.g. with a Cholesky dec.).
The columns of V are (the d-dimensional representatives of) x_0, x_1, g_0, g_1.

Step 2: build the worst-case function. By the interpolation theorem, there exists $f \in \mathcal{F}_{0,L}$ such that $f(x_i) = f_i$ and $\nabla f(x_i) = g_i$ for all i.
Take e.g. the *smallest such* interpolant.

Result: (f, x_0) is an explicit instance achieving

$$f(x_1) - f_\star = \tau^\star \|x_0 - x_\star\|^2.$$

No algorithm can do better on every function in $\mathcal{F}_{0,L}$.

In PEPit: call `problem.solve()` then access `.eval()` to obtain values and `f.get_interpolator()` to retrieve the worst-case function values automatically.

Dimension of the counter-example

The Gram matrix $G^\star \in \mathbb{R}^{n \times n}$ (where n = number of vectors, e.g. $n = 4$ for one GD step with function values).

Dimension of the counter-example

The Gram matrix $G^\star \in \mathbb{R}^{n \times n}$ (where n = number of vectors, e.g. $n = 4$ for one GD step with function values).

Rank of $G^\star$ = dimension of the worst-case example:

- In general, SDP solvers return full-rank $G^\star$ (high-dimensional).
- But worst-case examples often exist in **low dimension** (rank 1 or 2)!

Dimension of the counter-example

The Gram matrix $G^\star \in \mathbb{R}^{n \times n}$ (where n = number of vectors, e.g. $n = 4$ for one GD step with function values).

Rank of $G^\star$ = dimension of the worst-case example:

- In general, SDP solvers return full-rank $G^\star$ (high-dimensional).
- But worst-case examples often exist in **low dimension** (rank 1 or 2)!

$\rightsquigarrow$ **Rank minimisation heuristic** (`dimension_reduction_heuristic="trace"` in PEPit):

1. First solve the SDP to get $\tau^\star$.
2. Fix the objective, add constraint $\mathrm{Tr}(G\, M_{\mathrm{perf}}) = \tau^\star$.
3. Minimise $\mathrm{Tr}(G)$ (nuclear norm proxy for rank) subject to all constraints.
4. $\Rightarrow$ low-rank $G^\star$ $\Rightarrow$ **low-dimensional counter-example**.

Dimension of the counter-example

The Gram matrix $G^\star \in \mathbb{R}^{n \times n}$ (where n = number of vectors, e.g. $n = 4$ for one GD step with function values).

Rank of $G^\star$ = dimension of the worst-case example:

- In general, SDP solvers return full-rank $G^\star$ (high-dimensional).
- But worst-case examples often exist in **low dimension** (rank 1 or 2)!

$\rightsquigarrow$ **Rank minimisation heuristic** (`dimension_reduction_heuristic="trace"` in PEPit):

1. First solve the SDP to get $\tau^\star$.
2. Fix the objective, add constraint $\mathrm{Tr}(G\, M_{\text{perf}}) = \tau^\star$.
3. Minimise $\mathrm{Tr}(G)$ (nuclear norm proxy for rank) subject to all constraints.
4. $\Rightarrow$ low-rank $G^\star$ $\Rightarrow$ **low-dimensional counter-example**.

$\rightsquigarrow$ **Logdet heuristic** (`dimension_reduction_heuristic="logdet3"`)

Dimension of the counter-example

The Gram matrix $G^\star \in \mathbb{R}^{n \times n}$ (where n = number of vectors, e.g. $n = 4$ for one GD step with function values).

Rank of $G^\star$ = dimension of the worst-case example:

- In general, SDP solvers return full-rank $G^\star$ (high-dimensional).
- But worst-case examples often exist in **low dimension** (rank 1 or 2)!

$\rightsquigarrow$ **Rank minimisation heuristic** (`dimension_reduction_heuristic="trace"` in PEPit):

1. First solve the SDP to get $\tau^\star$.
2. Fix the objective, add constraint $\mathrm{Tr}(G\, M_{\text{perf}}) = \tau^\star$.
3. Minimise $\mathrm{Tr}(G)$ (nuclear norm proxy for rank) subject to all constraints.
4. $\Rightarrow$ low-rank $G^\star$ $\Rightarrow$ **low-dimensional counter-example**.

$\rightsquigarrow$ **Logdet heuristic** (`dimension_reduction_heuristic="logdet3"`)

Practical outcome

Simple low-dimensional functions often result in the worst case !

— Lab 1 —

Recovering counter example functions

Outline

From worst-case rate to worst-case instance

After `problem.solve()`, PEPit returns τ^* **and** the optimal Gram matrix $G^* \succcurlyeq 0$.

From worst-case rate to worst-case instance

After `problem.solve()`, PEPit returns τ^* **and** the optimal Gram matrix $G^* \succcurlyeq 0$.

What G^* encodes:

- $G^*_{ij} = \langle p_i, p_j \rangle$ for sampled vectors $p_i \in \{x_k - x_\star, g_k, \dots\}$.
- $\mathrm{rank}(G^*)$ = dimension of the worst-case instance.
- `point.eval()` extracts the coordinates.

From worst-case rate to worst-case instance

After `problem.solve()`, PEPit returns τ^* **and** the optimal Gram matrix $G^* \succcurlyeq 0$.

What G^* encodes:

- $G^*_{ij} = \langle p_i, p_j \rangle$ for sampled vectors $p_i \in \{x_k - x_\star, g_k, \dots\}$.
- $\mathrm{rank}(G^*)$ = dimension of the worst-case instance.
- `point.eval()` extracts the coordinates.

Rank-minimisation heuristic

Two-phase SDP:

1. Solve $\rightarrow \tau^*$.
2. Fix $\mathrm{Tr}(G\, M_{\mathrm{perf}}) = \tau^*$, minimise $\mathrm{Tr}(G)$ or $\log\det$.

$\Rightarrow$ low-rank G^*, *same* τ^*.

From worst-case rate to worst-case instance

After `problem.solve()`, PEPit returns τ^* **and** the optimal Gram matrix $G^* \succcurlyeq 0$.

What G^* encodes:

- $G^*_{ij} = \langle p_i, p_j \rangle$ for sampled vectors $p_i \in \{x_k - x_\star, g_k, \dots\}$.
- rank(G^*) = dimension of the worst-case instance.
- `point.eval()` extracts the coordinates.

Rank-minimisation heuristic

Two-phase SDP:

1. Solve $\rightarrow \tau^*$.
2. Fix $\mathrm{Tr}(G\, M_{\mathrm{perf}}) = \tau^*$, minimise $\mathrm{Tr}(G)$ or $\log\det$.

$\Rightarrow$ low-rank G^*, *same* τ^*.

```
pepit_tau = problem.solve(
    dimension_reduction_heuristic="logdet2",    # or "trace"
    tol_dimension_reduction=1e-6)
```

Example 1 – GD for Smooth Convex Minimization

Setting: $f \in \mathcal{F}_{0,L}$, $x_{t+1} = x_t - \gamma \nabla f(x_t)$, $\gamma = 1/L$, $n = 4$. Init: $\|x_0 - x_\star\|^2 \leq 1$. Perf: $f(x_n) - f_\star$.

Example 1 – GD for Smooth Convex Minimization

Setting: $f \in \mathcal{F}_{0,L}$, $x_{t+1} = x_t - \gamma \nabla f(x_t)$, $\gamma = 1/L$, $n = 4$. Init: $\|x_0 - x_\star\|^2 \leq 1$. Perf: $f(x_n) - f_\star$.

```python
from PEPit.functions import \
    SmoothStronglyConvexFunction

L = 1;  gamma = 1/L;  n = 4
problem = PEP()
f = problem.declare_function(
    SmoothStronglyConvexFunction, L=L, mu=0)

xs = f.stationary_point()
gs, fs = f.oracle(xs)
x0 = problem.set_initial_point()
gx, fx = f.oracle(x0)

problem.set_initial_condition((x0-xs)**2 <= 1)
for i in range(n):
    x0 = x0 - gamma * gx
    gx, fx = f.oracle(x0)
problem.set_performance_metric(fx - fs)

pepit_tau = problem.solve(
    dimension_reduction_heuristic="logdet2",
    tol_dimension_reduction=1e-6)
```

Example 1 – GD for Smooth Convex Minimization

Setting: $f \in \mathcal{F}_{0,L}$, $x_{t+1} = x_t - \gamma \nabla f(x_t)$, $\gamma = 1/L$, $n = 4$. Init: $\|x_0 - x_\star\|^2 \leq 1$. Perf: $f(x_n) - f_\star$.

```python
from PEPit.functions import \
    SmoothStronglyConvexFunction

L = 1;  gamma = 1/L;  n = 4
problem = PEP()
f = problem.declare_function(
    SmoothStronglyConvexFunction, L=L, mu=0)

xs = f.stationary_point()
gs, fs = f.oracle(xs)
x0 = problem.set_initial_point()
gx, fx = f.oracle(x0)

problem.set_initial_condition((x0-xs)**2 <= 1)
for i in range(n):
    x0 = x0 - gamma * gx
    gx, fx = f.oracle(x0)
problem.set_performance_metric(fx - fs)

pepit_tau = problem.solve(
    dimension_reduction_heuristic="logdet2",
    tol_dimension_reduction=1e-6)
```

Example 1 – GD for Smooth Convex Minimization

Setting: $f \in \mathcal{F}_{0,L}$, $x_{t+1} = x_t - \gamma \nabla f(x_t)$, $\gamma = 1/L$, $n = 4$. Init: $\|x_0 - x_\star\|^2 \leq 1$. Perf: $f(x_n) - f_\star$.

```python
from PEPit.functions import \
    SmoothStronglyConvexFunction

L = 1;  gamma = 1/L;  n = 4
problem = PEP()
f = problem.declare_function(
    SmoothStronglyConvexFunction, L=L, mu=0)

xs = f.stationary_point()
gs, fs = f.oracle(xs)
x0 = problem.set_initial_point()
gx, fx = f.oracle(x0)

problem.set_initial_condition((x0-xs)**2 <= 1)
for i in range(n):
    x0 = x0 - gamma * gx
    gx, fx = f.oracle(x0)
problem.set_performance_metric(fx - fs)

pepit_tau = problem.solve(
    dimension_reduction_heuristic="logdet2",
    tol_dimension_reduction=1e-6)
```

Gram matrix:

- **rank-1** $\Rightarrow$ **1D** worst-case
- Huber-loss-like function
- Can be verified analytically!

Example 1 – GD for Smooth Convex Minimization

Setting: $f \in \mathcal{F}_{0,L}$, $x_{t+1} = x_t - \gamma \nabla f(x_t)$, $\gamma = 1/L$, $n = 4$. Init: $\|x_0 - x_\star\|^2 \leq 1$. Perf: $f(x_n) - f_\star$.

```python
from PEPit.functions import \
    SmoothStronglyConvexFunction

L = 1;  gamma = 1/L;  n = 4
problem = PEP()
f = problem.declare_function(
    SmoothStronglyConvexFunction, L=L, mu=0)

xs = f.stationary_point()
gs, fs = f.oracle(xs)
x0 = problem.set_initial_point()
gx, fx = f.oracle(x0)

problem.set_initial_condition((x0-xs)**2 <= 1)
for i in range(n):
    x0 = x0 - gamma * gx
    gx, fx = f.oracle(x0)
problem.set_performance_metric(fx - fs)

pepit_tau = problem.solve(
    dimension_reduction_heuristic="logdet2",
    tol_dimension_reduction=1e-6)
```

Gram matrix:

- **rank-1** $\Rightarrow$ **1D** worst-case
- Huber-loss-like function
- Can be verified analytically!

Extract:

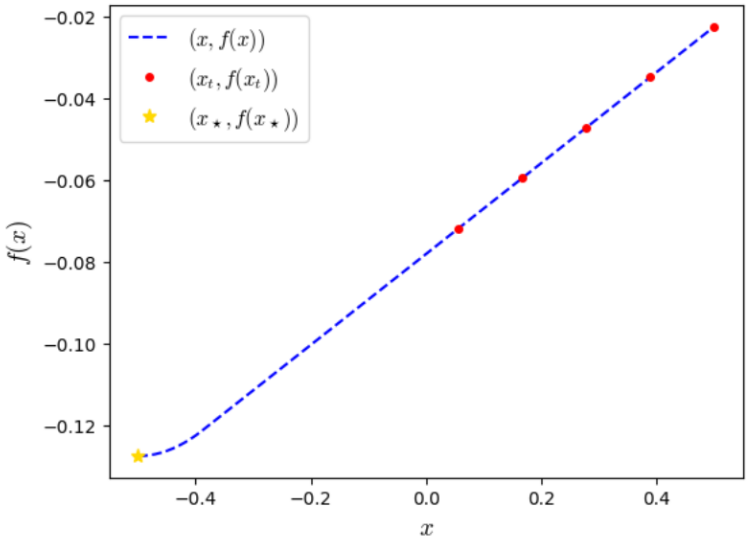
```
xs.eval()[0]
f.get_interpolator()
```

Example 1 – GD for Smooth Convex Minimization

```
(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) Setting up the problem: size of the Gram matrix: 7x7
(PEPit) Compiling SDP
(PEPit) Calling SDP solver
(PEPit) Solver status: optimal (wrapper:cvxpy, solver: SCS); optimal value: 0.05555881831011391
(PEPit) Postprocessing: 1 eigenvalue(s) > 1.6617368718060333e-06 before dimension reduction
(PEPit) Calling SDP solver
/usr/local/lib/python3.12/dist-packages/cvxpy/problems/problem.py:1510: UserWarning: Solution may be inaccurate. Try another solver, adjusting the solver
  warnings.warn(
(PEPit) Solver status: optimal_inaccurate (solver: SCS); objective value: 0.05555739775294698
(PEPit) Postprocessing: 1 eigenvalue(s) > 3.4965527908172573e-11 after 1 dimension reduction step(s)
(PEPit) Solver status: optimal_inaccurate (solver: SCS); objective value: 0.055557397696808924
(PEPit) Postprocessing: 1 eigenvalue(s) > 3.516410177191783e-11 after 2 dimension reduction step(s)
(PEPit) Solver status: optimal_inaccurate (solver: SCS); objective value: 0.055557397696808924
(PEPit) Postprocessing: 1 eigenvalue(s) > 3.516410177191783e-11 after dimension reduction
(PEPit) Postprocessing: solver's output is not entirely feasible (smallest eigenvalue of the Gram matrix is: -6.69e-07 < 0).
 Small deviation from 0 may simply be due to numerical error. Big ones should be deeply investigated.
 In any case, from now the provided values of parameters are based on the projection of the Gram matrix onto the cone of symmetric semi-definite matrix.
(PEPit) Primal feasibility check:
                The solver found a Gram matrix that is positive semi-definite up to an error of 6.693242677961387e-07
                All the primal scalar constraints are verified up to an error of 1.2427625796854191e-06
(PEPit) Dual feasibility check:
                The solver found a residual matrix that is positive semi-definite up to an error of 7.128579897617394e-17
                All the dual scalar values associated with inequality constraints are nonnegative
(PEPit) The worst-case guarantee proof is perfectly reconstituted up to an error of 3.4250635130243096e-06
(PEPit) Final upper bound (dual): 0.055555127602021075 and lower bound (primal example): 0.055557397696808924
(PEPit) Duality gap: absolute: -2.270094787849841e-06 and relative: -4.0860351311599134e-05
```

Figure: Pepit output

Example 1 – GD for Smooth Convex Minimization



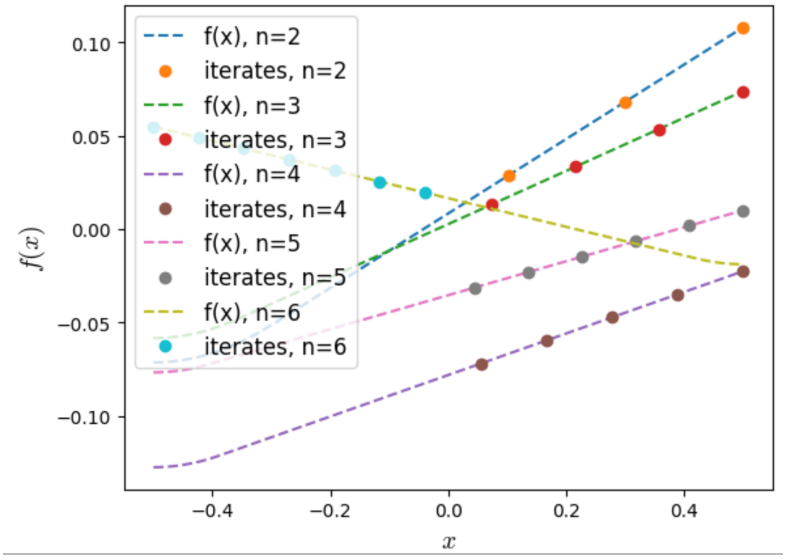

f(x), n=2
iterates, n=2
f(x), n=3
iterates, n=3
f(x), n=4
iterates, n=4
f(x), n=5
iterates, n=5
f(x), n=6
iterates, n=6
0.10
0.05
0.00
−0.05
−0.10
$f(x)$
−0.4
−0.2
0.0
0.2
0.4
x

Figure:

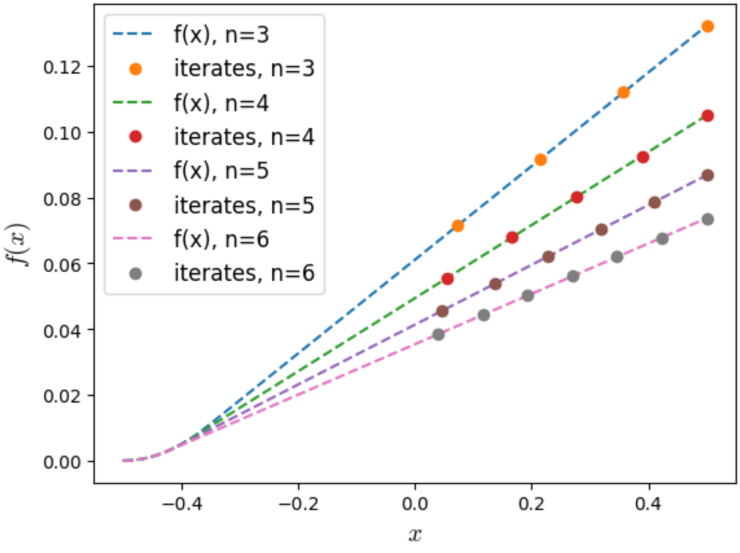

f(x), n=3
iterates, n=3
f(x), n=4
iterates, n=4
f(x), n=5
iterates, n=5
f(x), n=6
iterates, n=6
0.12
0.10
0.08
0.06
0.04
0.02
0.00
$f(x)$
−0.4
−0.2
0.0
0.2
0.4
x

Figure:

Example 2 – GD for Smooth (Non-Convex) Minimization

Setting: $f \in \mathcal{F}_{-L,L}$ (*no* convexity), $x_{t+1} = x_t - \gamma \nabla f(x_t)$, $\gamma = 0.95/L$, $n = 3$. Init: $f(x_0) - f_\star = 1$.
Perf: $\min_{0 \le t \le n} \|\nabla f(x_t)\|^2$.

Example 2 – GD for Smooth (Non-Convex) Minimization

Setting: $f \in \mathcal{F}_{-L,L}$ (*no* convexity), $x_{t+1} = x_t - \gamma \nabla f(x_t)$, $\gamma = 0.95/L$, $n = 3$. Init: $f(x_0) - f_\star = 1$.
Perf: $\min_{0 \le t \le n} \|\nabla f(x_t)\|^2$.

```
from PEPit.functions import SmoothFunction

L = 1;  n = 3;  gamma = .95/L
problem = PEP()
f = problem.declare_function(SmoothFunction, L=L)

xs = f.stationary_point()
gs, fs = f.oracle(xs)
x0 = problem.set_initial_point()
gx, fx = f.oracle(x0)

problem.set_initial_condition(f0 - fs == 1)
for i in range(n):
    problem.set_performance_metric(gx**2)
    f.add_constraint(fs <= fx - 1/(2*L) * gx**2)
    x0 = x0 - gamma * gx
    gx, fx = f.oracle(x0)
problem.set_performance_metric(gx**2)
f.add_constraint(fs <= fx - 1/(2*L) * gx**2)

pepit_tau = problem.solve(
    dimension_reduction_heuristic="logdet3",
    tol_dimension_reduction=1e-6)
```

Example 2 – GD for Smooth (Non-Convex) Minimization

Setting: $f \in \mathcal{F}_{-L,L}$ (*no* convexity), $x_{t+1} = x_t - \gamma \nabla f(x_t)$, $\gamma = 0.95/L$, $n = 3$. Init: $f(x_0) - f_\star = 1$.
Perf: $\min_{0 \le t \le n} \|\nabla f(x_t)\|^2$.

```python
from PEPit.functions import SmoothFunction

L = 1;  n = 3;  gamma = .95/L
problem = PEP()
f = problem.declare_function(SmoothFunction, L=L)

xs = f.stationary_point()
gs, fs = f.oracle(xs)
x0 = problem.set_initial_point()
gx, fx = f.oracle(x0)

problem.set_initial_condition(f0 - fs == 1)
for i in range(n):
    problem.set_performance_metric(gx**2)
    f.add_constraint(fs <= fx - 1/(2*L) * gx**2)
    x0 = x0 - gamma * gx
    gx, fx = f.oracle(x0)
problem.set_performance_metric(gx**2)
f.add_constraint(fs <= fx - 1/(2*L) * gx**2)

pepit_tau = problem.solve(
    dimension_reduction_heuristic="logdet3",
    tol_dimension_reduction=1e-6)
```

```
(PEPit) Setting up the problem: size of the Gram matrix: 6x6
(PEPit) Compiling SDP
(PEPit) Calling SDP solver
(PEPit) Solver status: optimal (wrapper:cvxpy, solver: SCS); optimal value: 0.3740940764949774 5
(PEPit) Postprocessing: 2 eigenvalue(s) > 3.3211383661205910e-06 before dimension reduction
(PEPit) Calling SDP solver
(PEPit) Solver status: optimal_inaccurate (solver: SCS); objective value: 0.3740935075590385
(PEPit) Postprocessing: 1 eigenvalue(s) > 0 after 1 dimension reduction step(s)
(PEPit) Solver status: optimal_inaccurate (solver: SCS); objective value: 0.3740934917166597 3
(PEPit) Postprocessing: 1 eigenvalue(s) > 0 after 2 dimension reduction step(s)
(PEPit) Solver status: optimal (solver: SCS); objective value: 0.3740934901113473
(PEPit) Postprocessing: 1 eigenvalue(s) > 9.055644664409157e-09 after 3 dimension reduction step(s)
(PEPit) Solver status: optimal (solver: SCS); objective value: 0.3740934901113473
(PEPit) Postprocessing: 1 eigenvalue(s) > 9.055644664409157e-09 after dimension reduction
```

Figure: Effectiveness of Dimred

Example 2 – GD for Smooth (Non-Convex) Minimization

Setting: $f \in \mathcal{F}_{-L,L}$ (*no* convexity), $x_{t+1} = x_t - \gamma \nabla f(x_t)$, $\gamma = 0.95/L$, $n = 3$. Init: $f(x_0) - f_\star = 1$.
Perf: $\min_{0 \le t \le n} \|\nabla f(x_t)\|^2$.

```python
from PEPit.functions import SmoothFunction

L = 1;  n = 3;  gamma = .95/L
problem = PEP()
f = problem.declare_function(SmoothFunction, L=L)

xs = f.stationary_point()
gs, fs = f.oracle(xs)
x0 = problem.set_initial_point()
gx, fx = f.oracle(x0)

problem.set_initial_condition(f0 - fs == 1)
for i in range(n):
    problem.set_performance_metric(gx**2)
    f.add_constraint(fs <= fx - 1/(2*L) * gx**2)
    x0 = x0 - gamma * gx
    gx, fx = f.oracle(x0)
problem.set_performance_metric(gx**2)
f.add_constraint(fs <= fx - 1/(2*L) * gx**2)

pepit_tau = problem.solve(
    dimension_reduction_heuristic="logdet3",
    tol_dimension_reduction=1e-6)
```

```
(PEPit) Setting up the problem: size of the Gram matrix: 6x6
(PEPit) Compiling SDP
(PEPit) Calling SDP solver
(PEPit) Solver status: optimal (wrapper:cvxpy, solver: SCS); optimal value: 0.37409407649497745
(PEPit) Postprocessing: 2 eigenvalue(s) > 3.3211383661205191e-06 before dimension reduction
(PEPit) Calling SDP solver
(PEPit) Solver status: optimal_inaccurate (solver: SCS); objective value: 0.3740935075590385
(PEPit) Postprocessing: 1 eigenvalue(s) > 0 after 1 dimension reduction step(s)
(PEPit) Solver status: optimal_inaccurate (solver: SCS); objective value: 0.37409349171665973
(PEPit) Postprocessing: 1 eigenvalue(s) > 0 after 2 dimension reduction step(s)
(PEPit) Solver status: optimal (solver: SCS); objective value: 0.3740934901113473
(PEPit) Postprocessing: 1 eigenvalue(s) > 9.855644664409157e-09 after 3 dimension reduction step(s)
(PEPit) Solver status: optimal (solver: SCS); objective value: 0.3740934901113473
(PEPit) Postprocessing: 1 eigenvalue(s) > 9.855644664409157e-09 after dimension reduction
```

Figure: Effectiveness of Dimred

- Multiple perf metrics: `set_performance_metric` keeps the *worst* over calls. $\inf_{\mathrm{Tr}(M_{perf,i}G) \le \tau} \tau$ is a convex problem.
- Oracle constraint: $f_\star \le f_t - \frac{1}{2L}\|g_t\|^2$ encodes that $x_\star$ is a stationary point no worse than any point in the trajectory and GD step from x_t.

Example 2 – GD for Smooth (Non-Convex) Minimization

Setting: $f \in \mathcal{F}_{-L,L}$ (*no* convexity), $x_{t+1} = x_t - \gamma \nabla f(x_t)$, $\gamma = 0.95/L$, $n = 3$. Init: $f(x_0) - f_\star = 1$.
Perf: $\min_{0 \le t \le n} \|\nabla f(x_t)\|^2$.

```python
from PEPit.functions import SmoothFunction

L = 1;  n = 3;  gamma = .95/L
problem = PEP()
f = problem.declare_function(SmoothFunction, L=L)

xs = f.stationary_point()
gs, fs = f.oracle(xs)
x0 = problem.set_initial_point()
gx, fx = f.oracle(x0)

problem.set_initial_condition(f0 - fs == 1)
for i in range(n):
    problem.set_performance_metric(gx**2)
    f.add_constraint(fs <= fx - 1/(2*L) * gx**2)
    x0 = x0 - gamma * gx
    gx, fx = f.oracle(x0)
problem.set_performance_metric(gx**2)
f.add_constraint(fs <= fx - 1/(2*L) * gx**2)

pepit_tau = problem.solve(
    dimension_reduction_heuristic="logdet3",
    tol_dimension_reduction=1e-6)
```

```
(PEPit) Setting up the problem: size of the Gram matrix: 6x6
(PEPit) Compiling SDP
(PEPit) Calling SDP solver
(PEPit) Solver status: optimal (wrapper:cvxpy, solver: SCS); optimal value: 0.37409407649497745
(PEPit) Postprocessing: 2 eigenvalue(s) > 3.3211383661205191e-06 before dimension reduction
(PEPit) Calling SDP solver
(PEPit) Solver status: optimal_inaccurate (solver: SCS); objective value: 0.3740935075590385
(PEPit) Postprocessing: 1 eigenvalue(s) > 0 after 1 dimension reduction step(s)
(PEPit) Solver status: optimal_inaccurate (solver: SCS); objective value: 0.37409349171665973
(PEPit) Postprocessing: 1 eigenvalue(s) > 0 after 2 dimension reduction step(s)
(PEPit) Solver status: optimal (solver: SCS); objective value: 0.3740934901113473
(PEPit) Postprocessing: 1 eigenvalue(s) > 9.855644664409157e-09 after 3 dimension reduction step(s)
(PEPit) Solver status: optimal (solver: SCS); objective value: 0.3740934901113473
(PEPit) Postprocessing: 1 eigenvalue(s) > 9.855644664409157e-09 after dimension reduction
```

Figure: Effectiveness of Dimred

- Multiple perf metrics: `set_performance_metric` keeps the *worst* over calls. $\inf_{\mathrm{Tr}(M_{perf,i}G) \le \tau} \tau$ is a convex problem.
- Oracle constraint: $f_\star \le f_t - \frac{1}{2L}\|g_t\|^2$ encodes that $x_\star$ is a stationary point no worse than any point in the trajectory and GD step from x_t.

Example 3 – GD with Exact Line Search on $\mathcal{F}_{\mu,L}$

Setting: $f \in \mathcal{F}_{\mu,L}$, $L = 1$, $\mu = 0.1$, $\gamma_k = \arg\min_{\gamma \geq 0} f(x_k - \gamma \nabla f(x_k))$, $n = 5$. Init: $f(x_0) - f_\star \leq 1$.
Perf: $f(x_n) - f_\star$.

Example 3 – GD with Exact Line Search on $\mathcal{F}_{\mu,L}$

Setting: $f \in \mathcal{F}_{\mu,L}$, $L = 1$, $\mu = 0.1$, $\gamma_k = \arg\min_{\gamma \geq 0} f(x_k - \gamma \nabla f(x_k))$, $n = 5$. Init: $f(x_0) - f_\star \leq 1$.
Perf: $f(x_n) - f_\star$.

```
from PEPit.functions import \
    SmoothStronglyConvexFunction
from PEPit.primitive_steps import \
    exact_linesearch_step

L, mu = 1, .1;  n = 5
problem = PEP()
f = problem.declare_function(
    SmoothStronglyConvexFunction, mu=mu, L=L)
xs = f.stationary_point()
gs, fs = f.oracle(xs)
x0 = problem.set_initial_point()
g0, f0 = f.oracle(x0)
problem.set_initial_condition(f0 - fs <= 1)

x = x0;  gx = g0
for i in range(n):
    x, gx, fx = exact_linesearch_step(x, f, [gx])

problem.set_performance_metric(fx - fs)
pepit_tau = problem.solve(
    dimension_reduction_heuristic='trace')

theoretical = ((L - mu)/(L + mu))**(2*n)
```

Example 3 – GD with Exact Line Search on $\mathcal{F}_{\mu,L}$

Setting: $f \in \mathcal{F}_{\mu,L}$, $L = 1$, $\mu = 0.1$, $\gamma_k = \arg\min_{\gamma \geq 0} f(x_k - \gamma \nabla f(x_k))$, $n = 5$. Init: $f(x_0) - f_\star \leq 1$. Perf: $f(x_n) - f_\star$.

```
from PEPit.functions import \
    SmoothStronglyConvexFunction
from PEPit.primitive_steps import \
    exact_linesearch_step

L, mu = 1, .1;  n = 5
problem = PEP()
f = problem.declare_function(
    SmoothStronglyConvexFunction, mu=mu, L=L)
xs = f.stationary_point()
gs, fs = f.oracle(xs)
x0 = problem.set_initial_point()
g0, f0 = f.oracle(x0)
problem.set_initial_condition(f0 - fs <= 1)

x = x0;  gx = g0
for i in range(n):
    x, gx, fx = exact_linesearch_step(x, f, [gx])

problem.set_performance_metric(fx - fs)
pepit_tau = problem.solve(
    dimension_reduction_heuristic='trace')

theoretical = ((L - mu)/(L + mu))**(2*n)
```

Results:

- $\tau_{\text{ls}}^* \approx 0.1344$
- GD fixed step: $\left(\frac{L-\mu}{L+\mu}\right)^{2n} = 0.134430$

→ **Line search has the same worst case performance as optimally tuned fixed step size GD.**

Example 3 – GD with Exact Line Search on $\mathcal{F}_{\mu,L}$

Setting: $f \in \mathcal{F}_{\mu,L}$, $L = 1$, $\mu = 0.1$, $\gamma_k = \arg\min_{\gamma \geq 0} f(x_k - \gamma \nabla f(x_k))$, $n = 5$. Init: $f(x_0) - f_\star \leq 1$. Perf: $f(x_n) - f_\star$.

```
from PEPit.functions import \
    SmoothStronglyConvexFunction
from PEPit.primitive_steps import \
    exact_linesearch_step

L, mu = 1, .1;  n = 5
problem = PEP()
f = problem.declare_function(
    SmoothStronglyConvexFunction, mu=mu, L=L)
xs = f.stationary_point()
gs, fs = f.oracle(xs)
x0 = problem.set_initial_point()
g0, f0 = f.oracle(x0)
problem.set_initial_condition(f0 - fs <= 1)

x = x0;  gx = g0
for i in range(n):
    x, gx, fx = exact_linesearch_step(x, f, [gx])

problem.set_performance_metric(fx - fs)
pepit_tau = problem.solve(
    dimension_reduction_heuristic='trace')

theoretical = ((L - mu)/(L + mu))**(2*n)
```

Results:

- $\tau_{\text{ls}}^* \approx 0.1344$
- GD fixed step: $\left(\frac{L-\mu}{L+\mu}\right)^{2n} = 0.134430$

→ **Line search has the same worst case performance as optimally tuned fixed step size GD.**

Gram matrix:

- **rank-2** ⇒ **2D** worst-case
- Iterates *zigzag* in $\mathbb{R}^2$
- Each step $\perp$ to previous gradient

Example 3 – GD with Exact Line Search on $\mathcal{F}_{\mu,L}$

Setting: $f \in \mathcal{F}_{\mu,L}$, $L = 1$, $\mu = 0.1$, $\gamma_k = \arg\min_{\gamma \geq 0} f(x_k - \gamma \nabla f(x_k))$, $n = 5$. Init: $f(x_0) - f_\star \leq 1$. Perf: $f(x_n) - f_\star$.

```python
from PEPit.functions import \
    SmoothStronglyConvexFunction
from PEPit.primitive_steps import \
    exact_linesearch_step

L, mu = 1, .1;  n = 5
problem = PEP()
f = problem.declare_function(
    SmoothStronglyConvexFunction, mu=mu, L=L)
xs = f.stationary_point()
gs, fs = f.oracle(xs)
x0 = problem.set_initial_point()
g0, f0 = f.oracle(x0)
problem.set_initial_condition(f0 - fs <= 1)

x = x0;  gx = g0
for i in range(n):
    x, gx, fx = exact_linesearch_step(x, f, [gx])

problem.set_performance_metric(fx - fs)
pepit_tau = problem.solve(
    dimension_reduction_heuristic='trace')

theoretical = ((L - mu)/(L + mu))**(2*n)
```

Results:

- $\tau_{\text{ls}}^* \approx 0.1344$
- GD fixed step: $\left(\frac{L-\mu}{L+\mu}\right)^{2n} = 0.134430$

→ **Line search has the same worst case performance as optimally tuned fixed step size GD.**

Gram matrix:

- **rank-2** ⇒ **2D** worst-case
- Iterates *zigzag* in $\mathbb{R}^2$
- Each step $\perp$ to previous gradient

Extract 2D:

```
x.eval()[0:2]
```

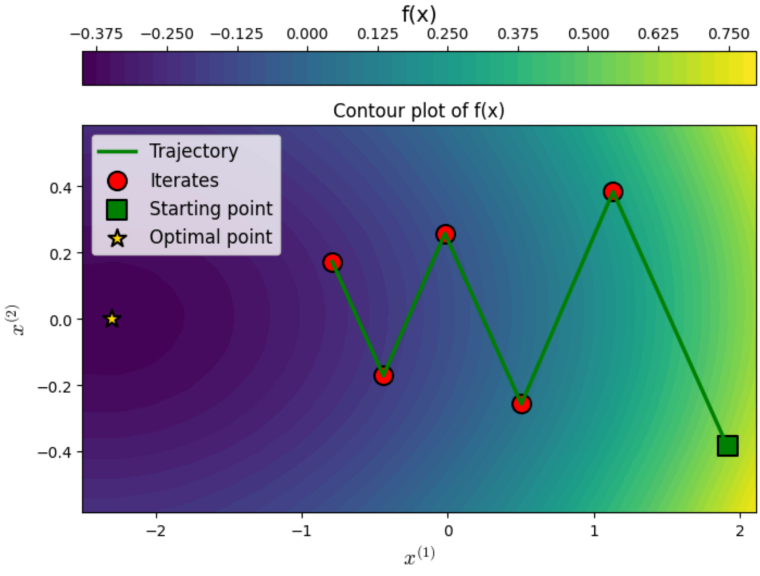

f(x)
−0.375
−0.250
−0.125
0.000
0.125
0.250
0.375
0.500
0.625
0.750
Contour plot of f(x)
Trajectory
Iterates
Starting point
Optimal point
0.4
0.2
0.0
−0.2
−0.4
$x^{(2)}$
−2
−1
0
1
2
$x^{(1)}$

Figure:

Take-Away: PEPit as a Counter-Example Generator

Example 1

GD, $\mathcal{F}_{0,L}$

Perf: $f(x_n) - f_\star$

rank-1 $\to$ 1D

Huber-loss function

Example 2

GD, $\mathcal{F}_{-L,L}$ non-convex

Perf: $\min_t \|\nabla f(x_t)\|^2$

rank-1 $\to$ 1D

non-convex, 1D profile

Example 3

GD + line search, $\mathcal{F}_{\mu,L}$

Perf: $f(x_n) - f_\star$

rank-2 $\to$ 2D

zigzag trajectory

Take-Away: PEPit as a Counter-Example Generator

Example 1	**Example 2**	**Example 3**
GD, $\mathcal{F}_{0,L}$	GD, $\mathcal{F}_{-L,L}$ non-convex	GD + line search, $\mathcal{F}_{\mu,L}$
Perf: $f(x_n) - f_\star$	Perf: $\min_t \|\nabla f(x_t)\|^2$	Perf: $f(x_n) - f_\star$
rank-1 $\to$ 1D	rank-1 $\to$ 1D	rank-2 $\to$ 2D
Huber-loss function	non-convex, 1D profile	zigzag trajectory

Three lines of interest:

1. `problem.solve(dimension_reduction_heuristic=...)` $\to \tau^*$ + low-rank G^*.
2. `point.eval()` $\to$ worst-case coordinates.
3. `f.get_interpolator()` $\to$ reconstruct worst-case function on a grid.

Take-Away: PEPit as a Counter-Example Generator

Example 1	**Example 2**	**Example 3**
GD, $\mathcal{F}_{0,L}$	GD, $\mathcal{F}_{-L,L}$ non-convex	GD + line search, $\mathcal{F}_{\mu,L}$
Perf: $f(x_n) - f_\star$	Perf: $\min_t \|\nabla f(x_t)\|^2$	Perf: $f(x_n) - f_\star$
rank-1 $\to$ 1D	rank-1 $\to$ 1D	rank-2 $\to$ 2D
Huber-loss function	non-convex, 1D profile	zigzag trajectory

Three lines of interest:

1. `problem.solve(dimension_reduction_heuristic=...)` $\to \tau^*$ + low-rank G^*.
2. `point.eval()` $\to$ worst-case coordinates.
3. `f.get_interpolator()` $\to$ reconstruct worst-case function on a grid.

$\to$ PEPit is not just a rate computer, it can produce *interpretable, verifiable* worst-case instances in low dimension.

Appendix