Performance Estimation Problems

A Tutorial – Session 1 / 3

March 2026

Aymeric Dieuleveut, Adrien Taylor

March 17, 2026

Outline

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:

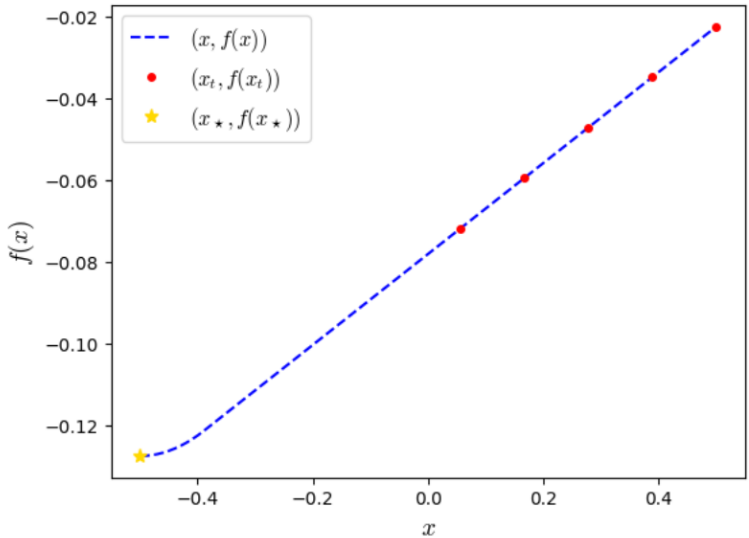
```
xs.eval()[0]
f.get_interpolator()
```

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



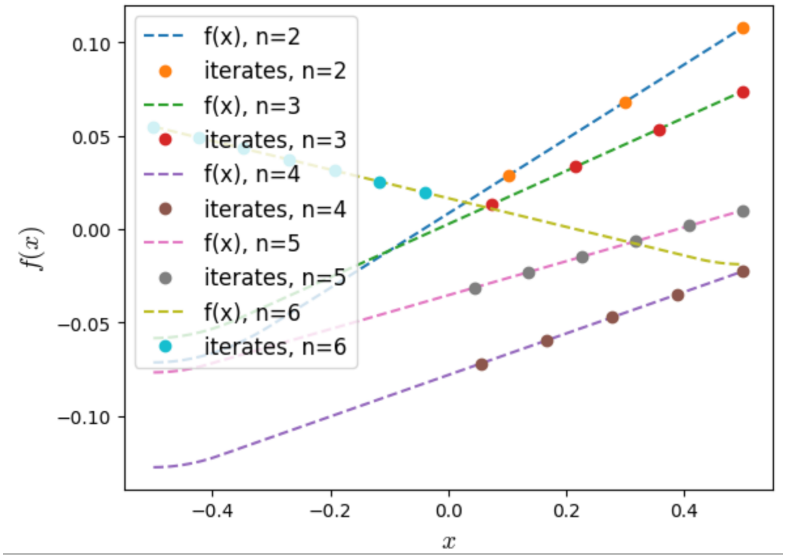

f(x), n=2
iterates, n=2
f(x), n=3
iterates, n=3
f(x), n=4
iterates, n=4
f(x), n=5
iterates, n=5
f(x), n=6
iterates, n=6
0.10
0.05
0.00
−0.05
−0.10
$f(x)$
−0.4
−0.2
0.0
0.2
0.4
x

Figure:

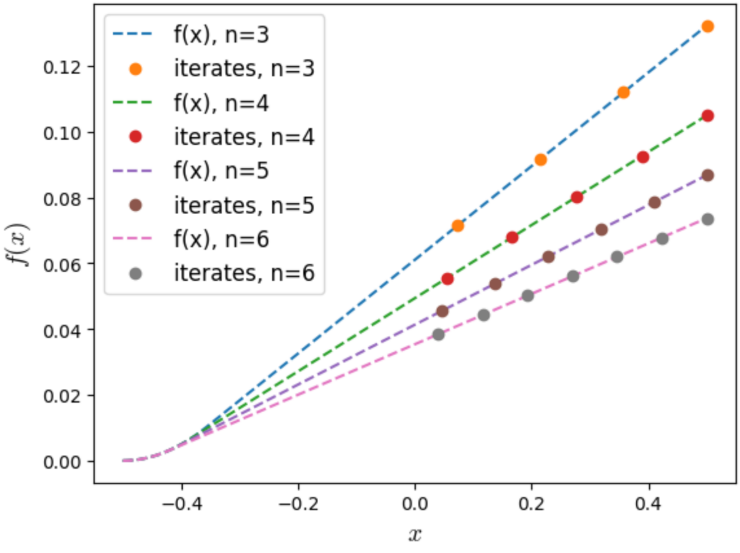

f(x), n=3
iterates, n=3
f(x), n=4
iterates, n=4
f(x), n=5
iterates, n=5
f(x), n=6
iterates, n=6
0.12
0.10
0.08
0.06
0.04
0.02
0.00
$f(x)$
−0.4
−0.2
0.0
0.2
0.4
x

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) > 3.3211383661205 91e-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) > 9.055644664409157e-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]
```

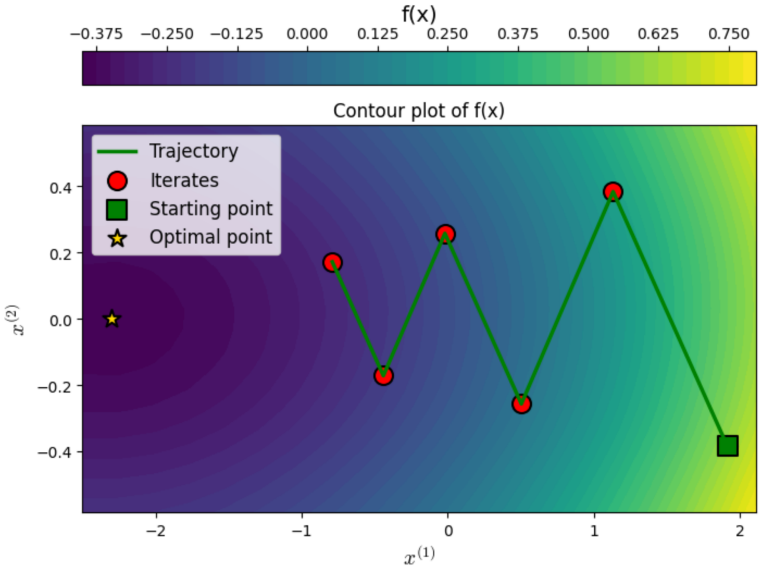

f(x)
−0.375 −0.250 −0.125 0.000 0.125 0.250 0.375 0.500 0.625 0.750
Contour plot of f(x)
Trajectory
Iterates
Starting point
Optimal point
0.4
0.2
0.0
−0.2
−0.4
$x^{(2)}$
−2 −1 0 1 2
$x^{(1)}$

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.

Outline

Interpolation: definition

Definition : Interpolation on a class $\mathcal{F}$

For $\mathcal{F} \subset \mathcal{C}(\mathbb{R}^d)$, a set of triplets $(x_i, g_i, f_i)_{i \in I}$ is **interpolable** by $\mathcal{F}$, written $\mathrm{Int}(\mathcal{F}, (x_i, g_i, f_i)_i)$, if

$$\exists f \in \mathcal{F} : \quad f(x_i) = f_i \ \text{ and } \ g_i \in \partial f(x_i) \quad \forall i \in I.$$

Interpolation: definition

Definition : Interpolation on a class $\mathcal{F}$

For $\mathcal{F} \subset \mathcal{C}(\mathbb{R}^d)$, a set of triplets $(x_i, g_i, f_i)_{i \in I}$ is **interpolable** by $\mathcal{F}$, written $\mathrm{Int}(\mathcal{F}, (x_i, g_i, f_i)_i)$, if

$$\exists f \in \mathcal{F} : \quad f(x_i) = f_i \ \text{ and } \ g_i \in \partial f(x_i) \quad \forall i \in I.$$

(Again) Where it matters. The constraint "$\exists f \in \mathcal{F}$" in the PEP is replaced by a **finite set of algebraic conditions** on the triplets.

Interpolation: definition

Definition : Interpolation on a class $\mathcal{F}$

For $\mathcal{F} \subset \mathcal{C}(\mathbb{R}^d)$, a set of triplets $(x_i, g_i, f_i)_{i \in I}$ is **interpolable** by $\mathcal{F}$, written $\mathrm{Int}(\mathcal{F}, (x_i, g_i, f_i)_i)$, if

$$\exists f \in \mathcal{F} : \quad f(x_i) = f_i \ \text{ and } \ g_i \in \partial f(x_i) \quad \forall i \in I.$$

(Again) Where it matters. The constraint "$\exists f \in \mathcal{F}$" in the PEP is replaced by a **finite set of algebraic conditions** on the triplets.

We derive these conditions for the three main classes below, building each one from the previous.

Interpolation: definition

Definition : Interpolation on a class $\mathcal{F}$

For $\mathcal{F} \subset \mathcal{C}(\mathbb{R}^d)$, a set of triplets $(x_i, g_i, f_i)_{i \in I}$ is **interpolable** by $\mathcal{F}$, written $\mathrm{Int}(\mathcal{F}, (x_i, g_i, f_i)_i)$, if

$$\exists f \in \mathcal{F} : \quad f(x_i) = f_i \ \text{ and } \ g_i \in \partial f(x_i) \quad \forall i \in I.$$

Theorem : Interpolation on convex functions

$\mathrm{Int}(\mathcal{F}_{0,\infty}, (x_i, g_i, f_i)_i) \iff$

Interpolation: definition

Definition : Interpolation on a class $\mathcal{F}$

For $\mathcal{F} \subset \mathcal{C}(\mathbb{R}^d)$, a set of triplets $(x_i, g_i, f_i)_{i \in I}$ is **interpolable** by $\mathcal{F}$, written $\mathrm{Int}(\mathcal{F}, (x_i, g_i, f_i)_i)$, if

$$\exists f \in \mathcal{F} : \quad f(x_i) = f_i \ \text{ and } \ g_i \in \partial f(x_i) \quad \forall i \in I.$$

Theorem : Interpolation on convex functions

$\mathrm{Int}(\mathcal{F}_{0,\infty}, (x_i, g_i, f_i)_i) \iff \forall i, j$:

$$f_i \geq f_j + \langle g_j, x_i - x_j \rangle.$$

Interpolation on $\mathcal{F}_{0,\infty}$ (convex functions)

Theorem : Interpolation on convex functions

$\mathrm{Int}(\mathcal{F}_{0,\infty}, (x_i, g_i, f_i)_i) \quad \Longleftrightarrow$

Interpolation on $\mathcal{F}_{0,\infty}$ (convex functions)

Theorem : Interpolation on convex functions

$\mathrm{Int}(\mathcal{F}_{0,\infty}, (x_i, g_i, f_i)_i) \iff \forall i, j$:

$$f_i \geq f_j + \langle g_j, x_i - x_j \rangle.$$

Interpolation on $\mathcal{F}_{0,\infty}$ (convex functions)

Theorem : Interpolation on convex functions

$\mathrm{Int}(\mathcal{F}_{0,\infty}, (x_i, g_i, f_i)_i) \iff \forall i, j:$

$$f_i \geq f_j + \langle g_j,\, x_i - x_j \rangle.$$

Proof sketch.

($\Rightarrow$) If f is convex, the subgradient inequality holds pointwise.

($\Leftarrow$) Build $f(x) = \max_j \{ f_j + \langle g_j, x - x_j \rangle \}$ (pointwise maximum of affine functions): convex, and interpolates by construction.

Interpolation on $\mathcal{F}_{0,\infty}$ (convex functions)

Theorem : Interpolation on convex functions

$\mathrm{Int}(\mathcal{F}_{0,\infty}, (x_i, g_i, f_i)_i) \iff \forall i, j:$

$$f_i \geq f_j + \langle g_j,\, x_i - x_j \rangle.$$

Proof sketch.

($\Rightarrow$) If f is convex, the subgradient inequality holds pointwise.

($\Leftarrow$) Build $f(x) = \max_j \{ f_j + \langle g_j, x - x_j \rangle \}$ (pointwise maximum of affine functions): convex, and interpolates by construction.

(again) Each condition is **linear in** f_i and **bilinear in** $(g_j, x_i - x_j)$
$\rightarrow$ after SDP lifting: $\mathrm{Tr}(G\, M_{ij}) + F^\top v_{ij} \geq 0$.

Interpolation on $\mathcal{F}_{\mu,\infty}$ (strongly convex)

Proposition : Interpolation on $\mathcal{F}_{\mu,\infty}$ (strongly convex)

$\mathrm{Int}(\mathcal{F}_{\mu,\infty}, (x_i, g_i, f_i)_i) \iff$

Interpolation on $\mathcal{F}_{\mu,\infty}$ (strongly convex)

Proposition : Interpolation on $\mathcal{F}_{\mu,\infty}$ (strongly convex)

$$\mathrm{Int}(\mathcal{F}_{\mu,\infty}, (x_i, g_i, f_i)_i) \iff \mathrm{Int}\left(\mathcal{F}_{0,\infty},\ \big(x_i,\ \underbrace{g_i - \mu x_i}_{\tilde{g}_i},\ \underbrace{f_i - \tfrac{\mu}{2}\|x_i\|^2}_{\tilde{f}_i}\big)_i\right)$$

Derivation. $f \in \mathcal{F}_{\mu,\infty}$ iff $h := f - \frac{\mu}{2}\|\cdot\|^2 \in \mathcal{F}_{0,\infty}$. Apply the convex interpolation condition to h:

$$\tilde{f}_i \geq \tilde{f}_j + \langle \tilde{g}_j, x_i - x_j\rangle \quad \Leftrightarrow \quad f_i - \tfrac{\mu}{2}\|x_i\|^2 \geq f_j - \tfrac{\mu}{2}\|x_j\|^2 + \langle g_j - \mu x_j,\ x_i - x_j\rangle.$$

Interpolation on $\mathcal{F}_{\mu,\infty}$ (strongly convex)

Proposition : Interpolation on $\mathcal{F}_{\mu,\infty}$ (strongly convex)

$$\mathrm{Int}(\mathcal{F}_{\mu,\infty}, (x_i, g_i, f_i)_i) \iff \mathrm{Int}\left(\mathcal{F}_{0,\infty},\ \big(x_i,\ \underbrace{g_i - \mu x_i}_{\tilde{g}_i},\ \underbrace{f_i - \tfrac{\mu}{2}\|x_i\|^2}_{\tilde{f}_i}\big)_i\right)$$

Derivation. $f \in \mathcal{F}_{\mu,\infty}$ iff $h := f - \frac{\mu}{2}\|\cdot\|^2 \in \mathcal{F}_{0,\infty}$. Apply the convex interpolation condition to h:

$$\tilde{f}_i \geq \tilde{f}_j + \langle \tilde{g}_j, x_i - x_j\rangle \quad \Leftrightarrow \quad f_i - \tfrac{\mu}{2}\|x_i\|^2 \geq f_j - \tfrac{\mu}{2}\|x_j\|^2 + \langle g_j - \mu x_j,\ x_i - x_j\rangle.$$

Interpolation on $\mathcal{F}_{\mu,\infty}$ (strongly convex)

Proposition : Interpolation on $\mathcal{F}_{\mu,\infty}$ (strongly convex)

$$\mathrm{Int}(\mathcal{F}_{\mu,\infty}, (x_i, g_i, f_i)_i) \iff \mathrm{Int}\left(\mathcal{F}_{0,\infty},\ \big(x_i,\ \underbrace{g_i - \mu x_i}_{\tilde{g}_i},\ \underbrace{f_i - \tfrac{\mu}{2}\|x_i\|^2}_{\tilde{f}_i}\big)_i\right)$$

Derivation. $f \in \mathcal{F}_{\mu,\infty}$ iff $h := f - \frac{\mu}{2}\|\cdot\|^2 \in \mathcal{F}_{0,\infty}$. Apply the convex interpolation condition to h:

$$\tilde{f}_i \geq \tilde{f}_j + \langle \tilde{g}_j, x_i - x_j\rangle \quad \Leftrightarrow \quad f_i - \tfrac{\mu}{2}\|x_i\|^2 \geq f_j - \tfrac{\mu}{2}\|x_j\|^2 + \langle g_j - \mu x_j,\ x_i - x_j\rangle.$$

Expanding and collecting:

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

(again) Each condition is quadratic in (x_i, g_i) and linear in f_i $\rightarrow$ SDP lifting works identically.

Interpolation on $\mathcal{F}_{0,L}$ (smooth convex): Fenchel duality

Proposition : Interpolation on $\mathcal{F}_{0,L}$

$\mathrm{Int}(\mathcal{F}_{0,L}, (x_i, g_i, f_i)_i) \iff$

Interpolation on $\mathcal{F}_{0,L}$ (smooth convex): Fenchel duality

Proposition : Interpolation on $\mathcal{F}_{0,L}$

$\mathrm{Int}(\mathcal{F}_{0,L}, (x_i, g_i, f_i)_i) \iff$

Key tool: Fenchel (convex) conjugate. For $f \in \mathcal{F}_{0,L}$, the conjugate $f^*(g) = \sup_x \{\langle g, x\rangle - f(x)\}$ satisfies:

Legendre–Fenchel facts:

- $f \in \mathcal{F}_{0,L} \;\Leftrightarrow\; f^* \in \mathcal{F}_{1/L,\infty}$
- $g_i = \nabla f(x_i) \;\Leftrightarrow\; x_i = \nabla f^*(g_i)$
- $f(x_i) + f^*(g_i) = \langle g_i, x_i\rangle$ (Fenchel–Young equality)

Interpolation on $\mathcal{F}_{0,L}$ (smooth convex): Fenchel duality

Proposition : Interpolation on $\mathcal{F}_{0,L}$

$\mathrm{Int}(\mathcal{F}_{0,L}, (x_i, g_i, f_i)_i) \iff$

Key tool: Fenchel (convex) conjugate. For $f \in \mathcal{F}_{0,L}$, the conjugate $f^*(g) = \sup_x \{\langle g, x\rangle - f(x)\}$ satisfies:

Legendre–Fenchel facts:

- $f \in \mathcal{F}_{0,L} \;\Leftrightarrow\; f^* \in \mathcal{F}_{1/L,\infty}$
- $g_i = \nabla f(x_i) \;\Leftrightarrow\; x_i = \nabla f^*(g_i)$
- $f(x_i) + f^*(g_i) = \langle g_i, x_i\rangle$ (Fenchel–Young equality)

If $f \in \mathcal{F}_{0,L}$, we define the *dual triplets*:

Interpolation on $\mathcal{F}_{0,L}$ (smooth convex): Fenchel duality

Proposition : Interpolation on $\mathcal{F}_{0,L}$

$\mathrm{Int}(\mathcal{F}_{0,L}, (x_i, g_i, f_i)_i) \iff$

Key tool: Fenchel (convex) conjugate. For $f \in \mathcal{F}_{0,L}$, the conjugate $f^*(g) = \sup_x \{\langle g, x\rangle - f(x)\}$ satisfies:

Legendre–Fenchel facts:

- $f \in \mathcal{F}_{0,L} \;\Leftrightarrow\; f^* \in \mathcal{F}_{1/L,\infty}$
- $g_i = \nabla f(x_i) \;\Leftrightarrow\; x_i = \nabla f^*(g_i)$
- $f(x_i) + f^*(g_i) = \langle g_i, x_i\rangle$ (Fenchel–Young equality)

If $f \in \mathcal{F}_{0,L}$, we define the *dual triplets*:

With $f_i^* := \langle g_i, x_i\rangle - f_i$
so that $f^* \in \mathcal{F}_{1/L,\infty}$ interpolates $(g_i, x_i, f_i^*)_i$.

Interpolation on $\mathcal{F}_{0,L}$ (smooth convex): Fenchel duality

Proposition : Interpolation on $\mathcal{F}_{0,L}$

$$\mathrm{Int}(\mathcal{F}_{0,L}, (x_i, g_i, f_i)_i) \iff \mathrm{Int}\big(\mathcal{F}_{1/L,\infty}, (g_i,\ x_i,\ \langle g_i, x_i\rangle - f_i)_i\big)$$

Key tool: Fenchel (convex) conjugate. For $f \in \mathcal{F}_{0,L}$, the conjugate $f^*(g) = \sup_x\{\langle g, x\rangle - f(x)\}$ satisfies:

Legendre–Fenchel facts:

- $f \in \mathcal{F}_{0,L} \ \Leftrightarrow\ f^* \in \mathcal{F}_{1/L,\infty}$
- $g_i = \nabla f(x_i) \ \Leftrightarrow\ x_i = \nabla f^*(g_i)$
- $f(x_i) + f^*(g_i) = \langle g_i, x_i\rangle$ (Fenchel–Young equality)

If $f \in \mathcal{F}_{0,L}$, we define the *dual triplets*:

With $f_i^* := \langle g_i, x_i\rangle - f_i$
so that $f^* \in \mathcal{F}_{1/L,\infty}$ interpolates $(g_i, x_i, f_i^*)_i$.

Interpolation on $\mathcal{F}_{0,L}$ (smooth convex): Fenchel duality

Proposition : Interpolation on $\mathcal{F}_{0,L}$

$$\mathrm{Int}(\mathcal{F}_{0,L}, (x_i, g_i, f_i)_i) \iff \mathrm{Int}\big(\mathcal{F}_{1/L,\infty}, (g_i,\ x_i,\ \langle g_i, x_i\rangle - f_i)_i\big)$$

Key tool: Fenchel (convex) conjugate. For $f \in \mathcal{F}_{0,L}$, the conjugate $f^*(g) = \sup_x\{\langle g, x\rangle - f(x)\}$ satisfies:

Legendre–Fenchel facts:

- $f \in \mathcal{F}_{0,L} \ \Leftrightarrow\ f^* \in \mathcal{F}_{1/L,\infty}$
- $g_i = \nabla f(x_i) \ \Leftrightarrow\ x_i = \nabla f^*(g_i)$
- $f(x_i) + f^*(g_i) = \langle g_i, x_i\rangle$ (Fenchel–Young equality)

If $f \in \mathcal{F}_{0,L}$, we define the *dual triplets*:

With $f_i^* := \langle g_i, x_i\rangle - f_i$
so that $f^* \in \mathcal{F}_{1/L,\infty}$ interpolates $(g_i, x_i, f_i^*)_i$.

Apply the $\mathcal{F}_{\mu,\infty}$ condition (with $\mu = 1/L$, variables (g_i, x_i, f_i^*)) and substitute $f_i^* = \langle g_i, x_i\rangle - f_i$:

$$f_i \ \geq\ f_j + \langle g_j,\ x_i - x_j\rangle + \tfrac{1}{2L}\|g_i - g_j\|^2. \qquad \text{(sampled co-coercivity)}$$

Remark: satisfying *both* sampled Lipschitz gradient $\left(\|g_i - g_j\|^2 \leq \frac{L^2}{\|} x_i - x_j\|^2\right)$ and sampled convexity separately $\big(f_i \geq f_j + \langle g_j, x_i - x_j\rangle\big)$ is **NOT sufficient** for interpolation.

Remark: satisfying *both* sampled Lipschitz gradient $\left(\|g_i - g_j\|^2 \leq L^2\|x_i - x_j\|^2\right)$ and sampled convexity separately $\left(f_i \geq f_j + \langle g_j, x_i - x_j\rangle\right)$ is **NOT sufficient** for interpolation.

Remark: satisfying *both* sampled Lipschitz gradient $\left(f_i \leq f_j + \langle g_j, x_i - x_j\rangle + \frac{L}{2}\|x_i - x_j\|^2\right)$ and sampled convexity separately $\left(f_i \geq f_j + \langle g_j, x_i - x_j\rangle\right)$ is **NOT sufficient** for interpolation.

Interpolation on $\mathcal{F}_{\mu,L}$ (smooth + strongly convex)

Combining both reductions (strong convexity and Fenchel duality):

Theorem : Interpolation on $\mathcal{F}_{\mu,L}$

$\mathrm{Int}(\mathcal{F}_{\mu,L}, (x_i, g_i, f_i)_i) \iff \forall i \neq j$:

$$f_i \geq f_j + \langle g_j,\, x_i - x_j \rangle + \tfrac{1}{2L}\|g_i - g_j\|^2 + \tfrac{\mu}{2(1-\mu/L)}\left\|x_i - x_j - \tfrac{g_i - g_j}{L}\right\|^2.$$

Interpolation on $\mathcal{F}_{\mu,L}$ (smooth + strongly convex)

Combining both reductions (strong convexity and Fenchel duality):

Theorem : Interpolation on $\mathcal{F}_{\mu,L}$

$\mathrm{Int}(\mathcal{F}_{\mu,L}, (x_i, g_i, f_i)_i) \iff \forall i \neq j$:

$$f_i \geq f_j + \langle g_j,\, x_i - x_j \rangle + \tfrac{1}{2L}\|g_i - g_j\|^2 + \tfrac{\mu}{2(1-\mu/L)}\left\|x_i - x_j - \tfrac{g_i - g_j}{L}\right\|^2.$$

Reading the condition:

- $\mu = 0$: recovers the $\mathcal{F}_{0,L}$ condition (smooth convex).
- $L = \infty$: recovers the $\mathcal{F}_{\mu,\infty}$ condition (strongly convex).
- Always: **one condition per ordered pair** (i, j), quadratic in (x, g), linear in f.

Interpolation on $\mathcal{F}_{\mu,L}$ (smooth + strongly convex)

Combining both reductions (strong convexity and Fenchel duality):

Theorem : Interpolation on $\mathcal{F}_{\mu,L}$

$\mathrm{Int}(\mathcal{F}_{\mu,L}, (x_i, g_i, f_i)_i) \iff \forall i \neq j$:

$$f_i \geq f_j + \langle g_j,\, x_i - x_j \rangle + \tfrac{1}{2L}\|g_i - g_j\|^2 + \tfrac{\mu}{2(1-\mu/L)}\left\|x_i - x_j - \tfrac{g_i - g_j}{L}\right\|^2.$$

Reading the condition:

$\mu = 0$: recovers the $\mathcal{F}_{0,L}$ condition (smooth convex).

$L = \infty$: recovers the $\mathcal{F}_{\mu,\infty}$ condition (strongly convex).

- Always: **one condition per ordered pair** (i, j), quadratic in (x, g), linear in f.

(again) As always, after SDP lifting: each condition becomes $\mathrm{Tr}(G\, M_{ij}) + F^\top v_{ij} \geq 0$ for explicit matrices $M_{ij} \in \mathbb{R}^{p \times p}$ and vectors $v_{ij} \in \mathbb{R}^q$.

Homogeneity: normalising the initial condition

Proposition : Homogeneity reduction

Suppose Perf and Init are **degree-2 homogeneous** in $(x_i, g_i)_i$ and **linear in** $()f(x_i)_i)$. Then:

$$\tau^* = \sup_{\substack{f\in\mathcal{F},\ \mathcal{A} \\ \mathrm{Init}(x_0)>0}} \frac{\mathrm{Perf}(x_T)}{\mathrm{Init}(x_0)} = \sup_{\substack{f\in\mathcal{F},\ \mathcal{A} \\ \mathrm{Init}(x_0)=1}} \mathrm{Perf}(x_T).$$

Homogeneity: normalising the initial condition

Proposition : Homogeneity reduction

Suppose Perf and Init are **degree-2 homogeneous** in $(x_i, g_i)_i$ and **linear in** $()f(x_i)_i)$. Then:

$$\tau^* = \sup_{\substack{f\in\mathcal{F},\ \mathcal{A} \\ \mathrm{Init}(x_0)>0}} \frac{\mathrm{Perf}(x_T)}{\mathrm{Init}(x_0)} = \sup_{\substack{f\in\mathcal{F},\ \mathcal{A} \\ \mathrm{Init}(x_0)=1}} \mathrm{Perf}(x_T).$$

Proof 1: **Via function scaling**

Fix any feasible instance (f, x_0) with $x_\star = 0$ and define the **scaled function**: $f_R(x) := R^2 f\left(\frac{x}{R}\right)$. Then:

1. $f_R \in \mathcal{F}_{\mu,L}$ (same class: L-smoothness and μ-convexity are invariant under this scaling).

Homogeneity: normalising the initial condition

Proposition : Homogeneity reduction

Suppose Perf and Init are **degree-2 homogeneous** in $(x_i, g_i)_i$ and **linear in** $()f(x_i)_i)$. Then:

$$\tau^* = \sup_{\substack{f\in\mathcal{F},\,\mathcal{A} \\ \mathrm{Init}(x_0)>0}} \frac{\mathrm{Perf}(x_T)}{\mathrm{Init}(x_0)} = \sup_{\substack{f\in\mathcal{F},\,\mathcal{A} \\ \mathrm{Init}(x_0)=1}} \mathrm{Perf}(x_T).$$

Proof 1: **Via function scaling**

Fix any feasible instance (f, x_0) with $x_\star = 0$ and define the **scaled function**: $f_R(x) := R^2 f\left(\frac{x}{R}\right)$. Then:

1. $f_R \in \mathcal{F}_{\mu,L}$ (same class: L-smoothness and μ-convexity are invariant under this scaling).
2. $\nabla f_R(x) = R\,\nabla f\left(\frac{x}{R}\right) \Rightarrow \nabla f_R(Rx) = R\,\nabla f\,(x)\,.$

Homogeneity: normalising the initial condition

Proposition : Homogeneity reduction

Suppose Perf and Init are **degree-2 homogeneous** in $(x_i, g_i)_i$ and **linear in** $()f(x_i)_i)$. Then:

$$\tau^* = \sup_{\substack{f\in\mathcal{F},\,\mathcal{A} \\ \mathrm{Init}(x_0)>0}} \frac{\mathrm{Perf}(x_T)}{\mathrm{Init}(x_0)} = \sup_{\substack{f\in\mathcal{F},\,\mathcal{A} \\ \mathrm{Init}(x_0)=1}} \mathrm{Perf}(x_T).$$

Proof 1: **Via function scaling**

Fix any feasible instance (f, x_0) with $x_\star = 0$ and define the **scaled function**: $f_R(x) := R^2 f\left(\frac{x}{R}\right)$. Then:

1. $f_R \in \mathcal{F}_{\mu,L}$ (same class: L-smoothness and μ-convexity are invariant under this scaling).
2. $\nabla f_R(x) = R\,\nabla f\left(\frac{x}{R}\right) \Rightarrow \nabla f_R(Rx) = R\,\nabla f\,(x)\,.$

$\Rightarrow$ Thus algorithm starting from $x_0^R = R\,x_0$ for f_R verifies $\forall t\ x_t^R = Rx_t$

Homogeneity: normalising the initial condition

Proposition : Homogeneity reduction

Suppose Perf and Init are **degree-2 homogeneous** in $(x_i, g_i)_i$ and **linear in** $()f(x_i)_i)$. Then:

$$\tau^* = \sup_{\substack{f\in\mathcal{F},\,\mathcal{A} \\ \mathrm{Init}(x_0)>0}} \frac{\mathrm{Perf}(x_T)}{\mathrm{Init}(x_0)} = \sup_{\substack{f\in\mathcal{F},\,\mathcal{A} \\ \mathrm{Init}(x_0)=1}} \mathrm{Perf}(x_T).$$

Proof 1: **Via function scaling**

Fix any feasible instance (f, x_0) with $x_\star = 0$ and define the **scaled function**: $f_R(x) := R^2 f\left(\frac{x}{R}\right)$. Then:

1. $f_R \in \mathcal{F}_{\mu,L}$ (same class: L-smoothness and μ-convexity are invariant under this scaling).
2. $\nabla f_R(x) = R\,\nabla f\left(\frac{x}{R}\right) \Rightarrow \nabla f_R(Rx) = R\,\nabla f\left(x\right).$

$\Rightarrow$ Thus algorithm starting from $x_0^R = R\,x_0$ for f_R verifies $\forall t\ x_t^R = Rx_t$

$\Rightarrow$ Thus the **ratio** is invariant,

$$\frac{\|x_1^R\|^2}{\|x_0^R\|^2} = \frac{R^2\|x_1\|^2}{R^2\|x_0\|^2} = \frac{\|x_1\|^2}{\|x_0\|^2},$$

so WLOG scale to $\|x_0\|^2 = 1$ by replacing $x_i \leftarrow x_i/R$, $g_i \leftarrow g_i/R$, $f_i \leftarrow f_i/R^2$.

Homogeneity: normalising the initial condition

Proposition : Homogeneity reduction

Suppose Perf and Init are **degree-2 homogeneous** in $(x_i, g_i)_i$ and **linear in** $()f(x_i)_i)$. Then:

$$\tau^* = \sup_{\substack{f \in \mathcal{F},\, \mathcal{A} \\ \mathrm{Init}(x_0) > 0}} \frac{\mathrm{Perf}(x_T)}{\mathrm{Init}(x_0)} = \sup_{\substack{f \in \mathcal{F},\, \mathcal{A} \\ \mathrm{Init}(x_0) = 1}} \mathrm{Perf}(x_T).$$

Proof 2: **After SDP / lifting**

Homogeneity: normalising the initial condition

Proposition : Homogeneity reduction

Suppose Perf and Init are **degree-2 homogeneous** in $(x_i, g_i)_i$ and **linear in** $()f(x_i)_i)$. Then:

$$\tau^* = \sup_{\substack{f\in\mathcal{F},\ \mathcal{A} \\ \mathrm{Init}(x_0)>0}} \frac{\mathrm{Perf}(x_T)}{\mathrm{Init}(x_0)} = \sup_{\substack{f\in\mathcal{F},\ \mathcal{A} \\ \mathrm{Init}(x_0)=1}} \mathrm{Perf}(x_T).$$

Proof 2: **After SDP / lifting**

After lifting, if we skip the homogenization step the PEP is:

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

Homogeneity: normalising the initial condition

Proposition : Homogeneity reduction

Suppose Perf and Init are **degree-2 homogeneous** in $(x_i, g_i)_i$ and **linear in** $()f(x_i)_i)$. Then:

$$\tau^* = \sup_{\substack{f\in\mathcal{F},\ \mathcal{A} \\ \mathrm{Init}(x_0)>0}} \frac{\mathrm{Perf}(x_T)}{\mathrm{Init}(x_0)} = \sup_{\substack{f\in\mathcal{F},\ \mathcal{A} \\ \mathrm{Init}(x_0)=1}} \mathrm{Perf}(x_T).$$

Proof 2: **After SDP / lifting**

After lifting, if we skip the homogenization step the PEP is:

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

$\rightarrow$ all constraints and the objective are **linear in** (G, F): (G, F) is feasible iif $(\lambda G, \lambda F)$ is.

Homogeneity: normalising the initial condition

Proposition : Homogeneity reduction

Suppose Perf and Init are **degree-2 homogeneous** in $(x_i, g_i)_i$ and **linear in** $()f(x_i)_i)$. Then:

$$\tau^* = \sup_{\substack{f\in\mathcal{F},\ \mathcal{A} \\ \mathrm{Init}(x_0)>0}} \frac{\mathrm{Perf}(x_T)}{\mathrm{Init}(x_0)} = \sup_{\substack{f\in\mathcal{F},\ \mathcal{A} \\ \mathrm{Init}(x_0)=1}} \mathrm{Perf}(x_T).$$

Proof 2: **After SDP / lifting**

After lifting, if we skip the homogenization step the PEP is:

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

→ all constraints and the objective are **linear in** (G, F): (G, F) is feasible iif $(\lambda G, \lambda F)$ is.

→ For any feasible point with a given objective value, we can build another feasible point , with the same objective value and $\mathrm{Tr}(G\,M_{\mathsf{init}}) + F^\top v_{init} = 1$.

Outline

Lagrangian Derivation of the Dual (no function values)

Primal (simplified: G only, no F):

$$\tau^* = \sup_{\substack{G \succcurlyeq 0,\ \mathrm{Tr}(G M_{\mathrm{init}}) = 1 \\ \forall k:\ \underbrace{\mathrm{Tr}(G\, M_k)}_{\mathrm{Ineq}_k} \leq 0}} \mathrm{Tr}(G\, M_{\mathrm{perf}})$$

Lagrangian Derivation of the Dual (no function values)

Primal (simplified: G only, no F):

$$\tau^* = \sup_{\substack{G \succcurlyeq 0,\ \mathrm{Tr}(G M_{\mathrm{init}})=1 \\ \forall k:\ \underbrace{\mathrm{Tr}(G\, M_k)}_{\mathrm{Ineq}_k} \leq 0}} \mathrm{Tr}(G\, M_{\mathrm{perf}})$$

Lagrangian (multipliers: $\tau \in \mathbb{R}$ for equality, $\lambda_k \geq 0$ for ≤ 0 constraints):

Lagrangian Derivation of the Dual (no function values)

Primal (simplified: G only, no F):

$$\tau^* = \sup_{\substack{G \succcurlyeq 0,\ \mathrm{Tr}(GM_{\text{init}})=1 \\ \forall k:\ \underbrace{\mathrm{Tr}(G\,M_k)}_{\text{Ineq}_k} \leq 0}} \mathrm{Tr}(G\,M_{\text{perf}})$$

Lagrangian (multipliers: $\tau \in \mathbb{R}$ for equality, $\lambda_k \geq 0$ for ≤ 0 constraints):

$$\mathcal{L}(G;\tau,\lambda) = \mathrm{Tr}(GM_{\text{perf}}) + \tau\big(1 - \mathrm{Tr}(GM_{\text{init}})\big) - \sum_k \lambda_k \mathrm{Tr}(GM_k) = \tau + \mathrm{Tr}\Big(G\ \underbrace{\big(M_{\text{perf}} - \tau M_{\text{init}} - \textstyle\sum_k \lambda_k M_k\big)}_{=:\,\Delta}\Big)$$

Lagrangian Derivation of the Dual (no function values)

Primal (simplified: G only, no F):

$$\tau^* = \sup_{\substack{G \succcurlyeq 0,\ \mathrm{Tr}(G M_{\mathrm{init}})=1 \\ \forall k:\ \underbrace{\mathrm{Tr}(G\, M_k) \le 0}_{\mathrm{Ineq}_k}}} \mathrm{Tr}(G\, M_{\mathrm{perf}})$$

Lagrangian (multipliers: $\tau \in \mathbb{R}$ for equality, $\lambda_k \ge 0$ for ≤ 0 constraints):

$$\mathcal{L}(G;\tau,\lambda) = \mathrm{Tr}(G M_{\mathrm{perf}}) + \tau\big(1 - \mathrm{Tr}(G M_{\mathrm{init}})\big) - \sum_k \lambda_k \mathrm{Tr}(G M_k) = \tau + \mathrm{Tr}\Big(G\ \underbrace{\big(M_{\mathrm{perf}} - \tau M_{\mathrm{init}} - \textstyle\sum_k \lambda_k M_k\big)}_{=:\,\Delta}\Big)$$

Dual function: $d(\tau,\lambda) = \sup_{G \succcurlyeq 0} \mathcal{L}(G;\tau,\lambda) = \tau + \sup_{G \succcurlyeq 0} \mathrm{Tr}(G\,\Delta)$.

Lagrangian Derivation of the Dual (no function values)

Primal (simplified: G only, no F):

$$\tau^* = \sup_{\substack{G \succcurlyeq 0,\ \mathrm{Tr}(G M_{\text{init}})=1 \\ \forall k:\ \underbrace{\mathrm{Tr}(G M_k)}_{\text{Ineq}_k} \le 0}} \mathrm{Tr}(G M_{\text{perf}})$$

Lagrangian (multipliers: $\tau \in \mathbb{R}$ for equality, $\lambda_k \ge 0$ for ≤ 0 constraints):

$$\mathcal{L}(G;\tau,\lambda) = \mathrm{Tr}(G M_{\text{perf}}) + \tau\big(1 - \mathrm{Tr}(G M_{\text{init}})\big) - \sum_k \lambda_k \mathrm{Tr}(G M_k) = \tau + \mathrm{Tr}\Big(G \underbrace{\big(M_{\text{perf}} - \tau M_{\text{init}} - \textstyle\sum_k \lambda_k M_k\big)}_{=:\,\Delta}\Big)$$

Dual function: $d(\tau,\lambda) = \sup_{G \succcurlyeq 0} \mathcal{L}(G;\tau,\lambda) = \tau + \sup_{G \succcurlyeq 0} \mathrm{Tr}(G\,\Delta)$.

$$\sup_{G \succcurlyeq 0} \mathrm{Tr}(G\,\Delta) = \begin{cases} 0 & \text{if } \Delta \preccurlyeq 0 \\ +\infty & \text{otherwise.} \end{cases} \qquad \text{Hence } d(\tau,\lambda) = \tau \text{ iff } \; M_{\text{perf}} - \tau M_{\text{init}} - \textstyle\sum_k \lambda_k M_k \preccurlyeq 0.$$

Lagrangian Derivation of the Dual (no function values)

Primal (simplified: G only, no F):

$$\tau^* = \sup_{\substack{G \succcurlyeq 0,\ \mathrm{Tr}(GM_{\text{init}})=1 \\ \forall k:\ \underbrace{\mathrm{Tr}(G\,M_k)}_{\text{Ineq}_k} \le 0}} \mathrm{Tr}(G\,M_{\text{perf}})$$

Lagrangian (multipliers: $\tau \in \mathbb{R}$ for equality, $\lambda_k \ge 0$ for ≤ 0 constraints):

$$\mathcal{L}(G;\tau,\lambda) = \mathrm{Tr}(GM_{\text{perf}}) + \tau\big(1 - \mathrm{Tr}(GM_{\text{init}})\big) - \sum_k \lambda_k \mathrm{Tr}(GM_k) = \tau + \mathrm{Tr}\Big(G\ \underbrace{\big(M_{\text{perf}} - \tau M_{\text{init}} - \textstyle\sum_k \lambda_k M_k\big)}_{=:\,\Delta}\Big)$$

Dual function: $d(\tau,\lambda) = \sup_{G\succcurlyeq 0} \mathcal{L}(G;\tau,\lambda) = \tau + \sup_{G\succcurlyeq 0} \mathrm{Tr}(G\,\Delta)$.

$$\sup_{G\succcurlyeq 0} \mathrm{Tr}(G\,\Delta) = \begin{cases} 0 & \text{if } \Delta \preccurlyeq 0 \\ +\infty & \text{otherwise.} \end{cases} \qquad \text{Hence } d(\tau,\lambda) = \tau \text{ iff } \ M_{\text{perf}} - \tau M_{\text{init}} - \textstyle\sum_k \lambda_k M_k \preccurlyeq 0.$$

Dual problem (minimise the upper bound):

$$\tau^* = \inf_{\substack{\lambda_k \ge 0,\ \tau \\ M_{\text{perf}} - \tau M_{\text{init}} - \sum_k \lambda_k M_k \preccurlyeq 0}} \tau$$

Lagrangian Derivation of the Dual (with function values F)

Full primal ($G \succcurlyeq 0$ and $F \in \mathbb{R}^p$):

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

Lagrangian Derivation of the Dual (with function values F)

Full primal ($G \succcurlyeq 0$ and $F \in \mathbb{R}^p$):

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

Lagrangian (same multipliers τ, $\lambda_k \geq 0$; now F is also a free variable):

$$\mathcal{L}(G, F; \tau, \lambda) = \mathrm{Tr}(GM_{\text{perf}}) + F^\top v_{\text{perf}} + \tau\big(1 - \mathrm{Tr}(GM_{\text{init}}) - F^\top v_{\text{init}}\big) - \sum_k \lambda_k \big(\mathrm{Tr}(GM_k) + F^\top v_k\big)$$

$$= \tau + \mathrm{Tr}\big(G\,\Delta_G\big) + F^\top \underbrace{\big(v_{\text{perf}} - \tau v_{\text{init}} - \textstyle\sum_k \lambda_k v_k\big)}_{=:\,\delta_F}$$

where $\Delta_G = M_{\text{perf}} - \tau M_{\text{init}} - \sum_k \lambda_k M_k$.

Lagrangian Derivation of the Dual (with function values F)

Full primal ($G \succcurlyeq 0$ and $F \in \mathbb{R}^p$):

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

Lagrangian (same multipliers τ, $\lambda_k \geq 0$; now F is also a free variable):

$$\mathcal{L}(G, F; \tau, \lambda) = \mathrm{Tr}(GM_{\text{perf}}) + F^\top v_{\text{perf}} + \tau\big(1 - \mathrm{Tr}(GM_{\text{init}}) - F^\top v_{\text{init}}\big) - \sum_k \lambda_k \big(\mathrm{Tr}(GM_k) + F^\top v_k\big)$$

$$= \tau + \mathrm{Tr}\big(G\,\Delta_G\big) + F^\top \underbrace{\big(v_{\text{perf}} - \tau v_{\text{init}} - \textstyle\sum_k \lambda_k v_k\big)}_{=:\,\delta_F}$$

where $\Delta_G = M_{\text{perf}} - \tau M_{\text{init}} - \sum_k \lambda_k M_k$.

$\sup_{G \succcurlyeq 0,\ F \in \mathbb{R}^p} \mathcal{L}(G, F; \tau, \lambda)$ is finite iff $\Delta_G \preccurlyeq 0$ **and** $\delta_F = 0$. When finite, it equals τ.

Lagrangian Derivation of the Dual (with function values F)

Full primal ($G \succcurlyeq 0$ and $F \in \mathbb{R}^p$):

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

Lagrangian (same multipliers τ, $\lambda_k \ge 0$; now F is also a free variable):

$$\mathcal{L}(G, F; \tau, \lambda) = \mathrm{Tr}(GM_{\text{perf}}) + F^\top v_{\text{perf}} + \tau\big(1 - \mathrm{Tr}(GM_{\text{init}}) - F^\top v_{\text{init}}\big) - \sum_k \lambda_k \big(\mathrm{Tr}(GM_k) + F^\top v_k\big)$$

$$= \tau + \mathrm{Tr}\big(G\,\Delta_G\big) + F^\top \underbrace{\big(v_{\text{perf}} - \tau v_{\text{init}} - \textstyle\sum_k \lambda_k v_k\big)}_{=:\,\delta_F}$$

where $\Delta_G = M_{\text{perf}} - \tau M_{\text{init}} - \sum_k \lambda_k M_k$.

$\sup_{G \succcurlyeq 0,\ F \in \mathbb{R}^p} \mathcal{L}(G, F; \tau, \lambda)$ is finite iff $\Delta_G \preccurlyeq 0$ **and** $\delta_F = 0$. When finite, it equals τ.

Full dual:

$$\tau^* = \inf_{\substack{\lambda_k \ge 0,\ \tau \\ M_{\text{perf}} - \tau M_{\text{init}} - \sum_k \lambda_k M_k \preccurlyeq 0 \\ v_{\text{perf}} - \tau v_{\text{init}} - \sum_k \lambda_k v_k = 0}} \tau$$

Dual Feasibility $\Rightarrow$ Valid Worst-Case Rate

Claim: if (τ, λ) is dual feasible, then τ is a valid worst-case rate, i.e. $\mathrm{Tr}(GM_{\mathrm{perf}}) \leq \tau$ for every primal feasible G.

Dual Feasibility $\Rightarrow$ Valid Worst-Case Rate

Claim: if (τ, λ) is dual feasible, then τ is a valid worst-case rate, i.e. $\mathrm{Tr}(GM_{\mathsf{perf}}) \leq \tau$ for every primal feasible G.

Weak duality: For any primal feasible G and dual feasible (τ, λ):

$$\mathrm{Tr}(GM_{\mathsf{perf}}) \leq \mathcal{L}(G; \tau, \lambda) \;\leq\; \tau.$$

(Proof)

$$\begin{aligned}\mathrm{Tr}(GM_{\mathsf{perf}}) &= \mathcal{L}(G; \tau, \lambda) - \tau\big(\underbrace{\mathrm{Tr}(GM_{\mathsf{init}})}_{=1} - 1\big) + \sum_k \lambda_k \underbrace{\mathrm{Tr}(GM_k)}_{\leq 0} \\ &= \tau + \mathrm{Tr}\Big(G \underbrace{\big(M_{\mathsf{perf}} - \tau M_{\mathsf{init}} - \textstyle\sum_k \lambda_k M_k\big)}_{=:\,\Delta}\Big) \leq \tau\end{aligned}$$

Dual Feasibility $\Rightarrow$ Valid Worst-Case Rate

Claim: if (τ, λ) is dual feasible, then τ is a valid worst-case rate, i.e. $\mathrm{Tr}(GM_{\mathrm{perf}}) \leq \tau$ for every primal feasible G.

Weak duality: For any primal feasible G and dual feasible (τ, λ):

$$\mathrm{Tr}(GM_{\mathrm{perf}}) \leq \mathcal{L}(G; \tau, \lambda) \;\leq\; \tau.$$

(Proof)

$$\begin{aligned}\mathrm{Tr}(GM_{\mathrm{perf}}) &= \mathcal{L}(G; \tau, \lambda) - \tau\big(\underbrace{\mathrm{Tr}(GM_{\mathrm{init}})}_{=1} - 1\big) + \sum_k \lambda_k \underbrace{\mathrm{Tr}(GM_k)}_{\leq 0} \\ &= \tau + \mathrm{Tr}\Big(G \underbrace{\big(M_{\mathrm{perf}} - \tau M_{\mathrm{init}} - \textstyle\sum_k \lambda_k M_k\big)}_{=:\,\Delta}\Big) \leq \tau\end{aligned}$$

Consequence. Any dual feasible point (τ, λ) provides:

- $\rightarrow$ A **valid upper bound** τ on the worst-case performance.
- $\rightarrow$ An **explicit proof certificate**: the multipliers λ_k identify which inequalities matter.

Strong duality (Slater) $\Rightarrow$ the dual optimum equals τ^*.

The LMI as a Proof of the Rate

Setup: dual feasible $(\tau, \lambda_k \geq 0)$ with $M_{\text{perf}} - \tau M_{\text{init}} - \sum_k \lambda_k M_k \preccurlyeq 0$.

The LMI as a Proof of the Rate

Setup: dual feasible $(\tau, \lambda_k \geq 0)$ with $M_{\text{perf}} - \tau M_{\text{init}} - \sum_k \lambda_k M_k \preccurlyeq 0$.

Step 1 – LMI $\Rightarrow$ inequality for all $G \succcurlyeq 0$:

$$\forall G \succcurlyeq 0: \quad \mathrm{Tr}\big(G\,(M_{\text{perf}} - \tau M_{\text{init}} - \textstyle\sum_k \lambda_k M_k)\big) \leq 0$$

$$(G \succcurlyeq 0,\ \text{LMI} \preccurlyeq 0 \Rightarrow \mathrm{Tr}(G \cdot \text{LMI}) \leq 0)$$

The LMI as a Proof of the Rate

Setup: dual feasible $(\tau, \lambda_k \geq 0)$ with $M_{\text{perf}} - \tau M_{\text{init}} - \sum_k \lambda_k M_k \preccurlyeq 0$.

Step 1 – LMI $\Rightarrow$ inequality for all $G \succcurlyeq 0$:

$$\forall G \succcurlyeq 0: \quad \operatorname{Tr}\big(G\,(M_{\text{perf}} - \tau M_{\text{init}} - \textstyle\sum_k \lambda_k M_k)\big) \leq 0$$

$$(G \succcurlyeq 0,\ \text{LMI} \preccurlyeq 0 \Rightarrow \operatorname{Tr}(G \cdot \text{LMI}) \leq 0)$$

Step 2 – Rewrite for all problem instances (all $G \succcurlyeq 0$):

$$\underbrace{\operatorname{Tr}(GM_{\text{perf}})}_{\text{Perf}(x_T)} \leq \tau \underbrace{\operatorname{Tr}(GM_{\text{init}})}_{\text{Init}(x_0)} + \sum_k \lambda_k \underbrace{\operatorname{Tr}(GM_k)}_{\text{Ineq}_k}$$

The LMI as a Proof of the Rate

Setup: dual feasible $(\tau, \lambda_k \geq 0)$ with $M_{\text{perf}} - \tau M_{\text{init}} - \sum_k \lambda_k M_k \preccurlyeq 0$.

Step 1 – LMI $\Rightarrow$ inequality for all $G \succcurlyeq 0$:

$$\forall G \succcurlyeq 0: \quad \operatorname{Tr}\big(G\,(M_{\text{perf}} - \tau M_{\text{init}} - \textstyle\sum_k \lambda_k M_k)\big) \leq 0$$

$$(G \succcurlyeq 0,\ \text{LMI} \preccurlyeq 0 \Rightarrow \operatorname{Tr}(G \cdot \text{LMI}) \leq 0)$$

Step 2 – Rewrite for all problem instances (all $G \succcurlyeq 0$):

$$\underbrace{\operatorname{Tr}(GM_{\text{perf}})}_{\text{Perf}(x_T)} \leq \tau \underbrace{\operatorname{Tr}(GM_{\text{init}})}_{\text{Init}(x_0)} + \sum_k \lambda_k \underbrace{\operatorname{Tr}(GM_k)}_{\text{Ineq}_k}$$

Step 3 – Use algorithm & function-class constraints:
Under the algorithm and function class $\Rightarrow \text{Ineq}_k \leq 0$ for all k, and $\lambda_k \geq 0$, so $\sum_k \lambda_k \text{Ineq}_k \leq 0$:

$$\text{Perf}(x_T) \leq \tau \cdot \text{Init}(x_0) + \underbrace{\sum_k \lambda_k \overbrace{\text{Ineq}_k}^{\leq 0 \text{ (algo + class)}}}_{\leq 0}$$

The LMI as a Proof of the Rate

Setup: dual feasible $(\tau, \lambda_k \geq 0)$ with $M_{\text{perf}} - \tau M_{\text{init}} - \sum_k \lambda_k M_k \preccurlyeq 0$.

Step 1 – LMI $\Rightarrow$ inequality for all $G \succcurlyeq 0$:

$$\forall G \succcurlyeq 0: \quad \operatorname{Tr}\big(G\,(M_{\text{perf}} - \tau M_{\text{init}} - \textstyle\sum_k \lambda_k M_k)\big) \leq 0$$

$$(G \succcurlyeq 0,\ \text{LMI} \preccurlyeq 0 \Rightarrow \operatorname{Tr}(G \cdot \text{LMI}) \leq 0)$$

Step 2 – Rewrite for all problem instances (all $G \succcurlyeq 0$):

$$\underbrace{\operatorname{Tr}(GM_{\text{perf}})}_{\text{Perf}(x_T)} \leq \tau \underbrace{\operatorname{Tr}(GM_{\text{init}})}_{\text{Init}(x_0)} + \sum_k \lambda_k \underbrace{\operatorname{Tr}(GM_k)}_{\text{Ineq}_k}$$

Step 3 – Use algorithm & function-class constraints:
Under the algorithm and function class $\Rightarrow \text{Ineq}_k \leq 0$ for all k, and $\lambda_k \geq 0$, so $\sum_k \lambda_k \text{Ineq}_k \leq 0$:

$$\text{Perf}(x_T) \leq \tau \cdot \text{Init}(x_0) + \underbrace{\sum_k \lambda_k \overbrace{\text{Ineq}_k}^{\leq 0 \text{ (algo + class)}}}_{\leq 0}$$

$$\text{Perf}(x_T) \leq \tau \cdot \text{Init}(x_0).$$

The LMI is a formal proof of the rate τ. The λ_k are its multipliers.

Primal vs Dual – summary – two sides of the same coin

PEP-primal (find worst-case instance):

$$\tau^* = \sup_{\substack{0 \preccurlyeq G \\ \mathrm{Tr}(G\, M_{\mathrm{init}})=1 \\ \forall k,\, \mathrm{Tr}(G\, M_k) \geq 0}} \mathrm{Tr}(G\, M_{\mathrm{perf}})$$

- Primal feasible $G \Rightarrow$ lower bound on rate
- Primal optimal $G^\star \Rightarrow$ tight counter-example

Primal vs Dual – summary – two sides of the same coin

PEP-primal (find worst-case instance):

$$\tau^* = \sup_{\substack{0 \preccurlyeq G \\ \mathrm{Tr}(G\, M_{\mathrm{init}})=1 \\ \forall k,\ \mathrm{Tr}(G\, M_k)\geq 0}} \mathrm{Tr}(G\, M_{\mathrm{perf}})$$

- Primal feasible $G \Rightarrow$ lower bound on rate
- Primal optimal $G^\star \Rightarrow$ tight counter-example

PEP-dual (find proof certificate):

$$\tau^* = \inf_{\substack{(\lambda_k)\geq 0,\ \tau\geq 0 \\ M_{\mathrm{perf}} - \tau\, M_{\mathrm{init}} - \sum_k \lambda_k M_k \preccurlyeq 0}} \tau$$

- Dual feasible $(\lambda_k), \tau \Rightarrow$ upper bound on rate
- Dual optimal $(\lambda_k^\star), \tau^\star \Rightarrow$ explicit proof (weighted sum of interpolation inequalities)

Primal vs Dual – summary – two sides of the same coin

PEP-primal (find worst-case instance):

$$\tau^* = \sup_{\substack{0 \preccurlyeq G \\ \mathrm{Tr}(G\,M_{\text{init}})=1 \\ \forall k,\ \mathrm{Tr}(G\,M_k)\geq 0}} \mathrm{Tr}(G\,M_{\text{perf}})$$

- Primal feasible $G \Rightarrow$ lower bound on rate
- Primal optimal $G^\star \Rightarrow$ tight counter-example

PEP-dual (find proof certificate):

$$\tau^* = \inf_{\substack{(\lambda_k)\geq 0,\ \tau\geq 0 \\ M_{\text{perf}} - \tau\, M_{\text{init}} - \sum_k \lambda_k M_k \preccurlyeq 0}} \tau$$

- Dual feasible $(\lambda_k), \tau \Rightarrow$ upper bound on rate
- Dual optimal $(\lambda_k^\star), \tau^\star \Rightarrow$ explicit proof (weighted sum of interpolation inequalities)

Strong duality (Slater's condition): $\tau^*_{\text{primal}} = \tau^*_{\text{dual}}$.

A rate τ is valid if and only if $\exists\,(\lambda_k) \geq 0 :\ M_{\text{perf}} - \tau\, M_{\text{init}} - \sum_k \lambda_k M_k \preccurlyeq 0$.

Interactive Session #3 (Easy)

Exploring the **dual variables** with the Streamlit UI

Activity: GD on $\mathcal{F}_{\mu,L}$ — explore dual multipliers for $\|x_1 - x_\star\|^2$.

1. Go to `pepitui-test.streamlit.app`
2. Select **Gradient Descent**, performance metric $\|x_1 - x_\star\|^2$, init $\|x_0 - x_\star\|^2 \leq 1$
3. Run `Solve` and observe τ^*
4. Select **both** interpolation constraint types ($0 \to \star$ and $\star \to 0$) and plot them as a function of γ.
5. Why don't we plot the dual value associated wih the init constraint?
6. Try to guess the symbolic value of the dual values

Interactive Session #4 (Advanced)

Dual variables and **minimal proof certificates**

Activity: GD on $\mathcal{F}_{0,L}$ — dual certificate for $f(x_1) - f_\star$.

1. Switch to performance metric $f(x_1) - f_\star$
2. *Why are there more interpolation points involved?*
3. Run `Solve`,
4. Look at all **dual** multipliers λ_{ij}

$\Rightarrow$ **Play with `Recompute`:**

- Select / unselect individual interpolation inequalities and recompute — how does τ change?
- Can you obtain a *larger* or *smaller* value by removing constraints?
- **What is the minimal set of inequalities for which the tightest rate is still achieved?**

GD on $\mathcal{F}_{\mu,L}$: the 2×2 Gram Matrix

Setting: $f \in \mathcal{F}_{\mu,L}$, $x_1 = x_0 - \gamma g_0$, $g_0 = \nabla f(x_0)$, $g_\star = 0$. Init: $\|x_0 - x_\star\|^2 \leq 1$. Perf: $\|x_1 - x_\star\|^2$.

GD on $\mathcal{F}_{\mu,L}$: the 2×2 Gram Matrix

Setting: $f \in \mathcal{F}_{\mu,L}$, $x_1 = x_0 - \gamma g_0$, $g_0 = \nabla f(x_0)$, $g_\star = 0$. Init: $\|x_0 - x_\star\|^2 \le 1$. Perf: $\|x_1 - x_\star\|^2$.

Sampled vectors: $p_1 = x_0 - x_\star$, $p_2 = g_0$. Function values: $F = \begin{pmatrix} f_0 \\ f_\star \end{pmatrix}$.

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

GD on $\mathcal{F}_{\mu,L}$: the 2×2 Gram Matrix

Setting: $f \in \mathcal{F}_{\mu,L}$, $x_1 = x_0 - \gamma g_0$, $g_0 = \nabla f(x_0)$, $g_\star = 0$. Init: $\|x_0 - x_\star\|^2 \leq 1$. Perf: $\|x_1 - x_\star\|^2$.

Sampled vectors: $p_1 = x_0 - x_\star$, $p_2 = g_0$. Function values: $F = \begin{pmatrix} f_0 \\ f_\star \end{pmatrix}$.

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

Encoding matrices (no F term for Perf and Init):

$$M_{\text{perf}} = \begin{pmatrix} 1 & -\gamma \\ -\gamma & \gamma^2 \end{pmatrix} \text{ encodes } \|x_1 - x_\star\|^2, \qquad M_{\text{init}} = \begin{pmatrix} 1 & 0 \\ 0 & 0 \end{pmatrix} \text{ encodes } \|x_0 - x_\star\|^2.$$

GD on $\mathcal{F}_{\mu,L}$: the 2×2 Gram Matrix

Setting: $f \in \mathcal{F}_{\mu,L}$, $x_1 = x_0 - \gamma g_0$, $g_0 = \nabla f(x_0)$, $g_\star = 0$. Init: $\|x_0 - x_\star\|^2 \le 1$. Perf: $\|x_1 - x_\star\|^2$.

Sampled vectors: $p_1 = x_0 - x_\star$, $p_2 = g_0$. Function values: $F = \begin{pmatrix} f_0 \\ f_\star \end{pmatrix}$.

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

Encoding matrices (no F term for Perf and Init):

$$M_{\text{perf}} = \begin{pmatrix} 1 & -\gamma \\ -\gamma & \gamma^2 \end{pmatrix} \text{ encodes } \|x_1 - x_\star\|^2, \qquad M_{\text{init}} = \begin{pmatrix} 1 & 0 \\ 0 & 0 \end{pmatrix} \text{ encodes } \|x_0 - x_\star\|^2.$$

$$\tau^* = \sup_{f_0, f_\star,\ G \succcurlyeq 0} \operatorname{Tr}(M_{\text{perf}}\, G) \quad \text{s.t.} \quad \begin{cases} f_0 - f_\star + \operatorname{Tr}(M_{0\star} G) \le 0 & [(0,\star) \text{ interp.}] \\ f_\star - f_0 + \operatorname{Tr}(M_{\star 0} G) \le 0 & [(\star,0) \text{ interp.}] \\ \operatorname{Tr}(M_{\text{init}}\, G) \le 1 & [\text{normalization}] \end{cases}$$

Interpolation Constraints: Matrices M_{ij} and Vectors v_{ij}

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

Interpolation Constraints: Matrices M_{ij} and Vectors v_{ij}

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

Condition $(0, \star)$:

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

$$M_{0\star} = \frac{1}{2(L-\mu)} \begin{pmatrix} \mu L & -L \\ -L & 1 \end{pmatrix}, \quad v_{0\star} = \begin{pmatrix} 1 \\ -1 \end{pmatrix}$$

i.e. $f_0 - f_\star + \mathrm{Tr}(M_{0\star}\, G) \leq 0$,
encodes
$f_\star \geq f_0 + \langle g_0, x_\star - x_0 \rangle + \frac{1}{2L}\|g_0\|^2 + \frac{\mu}{2(1-\mu/L)}\|x_0 - x_\star - \frac{g_0}{L}\|^2$

Interpolation Constraints: Matrices M_{ij} and Vectors v_{ij}

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

Condition $(0, \star)$:

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

$$M_{0\star} = \frac{1}{2(L-\mu)} \begin{pmatrix} \mu L & -L \\ -L & 1 \end{pmatrix}, \quad v_{0\star} = \begin{pmatrix} 1 \\ -1 \end{pmatrix}$$

i.e. $f_0 - f_\star + \mathrm{Tr}(M_{0\star}\, G) \leq 0$,
encodes
$f_\star \geq f_0 + \langle g_0, x_\star - x_0 \rangle + \frac{1}{2L}\|g_0\|^2 + \frac{\mu}{2(1-\mu/L)}\|x_0 - x_\star - \frac{g_0}{L}\|^2$

Condition $(\star, 0)$:

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

$$M_{\star 0} = \frac{1}{2(L-\mu)} \begin{pmatrix} \mu L & -\mu \\ -\mu & 1 \end{pmatrix}, \quad v_{\star 0} = \begin{pmatrix} -1 \\ 1 \end{pmatrix}$$

i.e. $f_\star - f_0 + \mathrm{Tr}(M_{\star 0}\, G) \leq 0$,
encodes $f_0 \geq f_\star + \frac{1}{2L}\|g_0\|^2 + \frac{\mu}{2(1-\mu/L)}\|x_0 - x_\star - \frac{g_0}{L}\|^2$

Dual: Multipliers, $\lambda_1 = \lambda_2$, and the Explicit LMI

Associate one multiplier per constraint:

$$f_0 - f_\star + \mathrm{Tr}(M_{0\star} G) \le 0 : \lambda_1, \qquad f_\star - f_0 + \mathrm{Tr}(M_{\star 0} G) \le 0 : \lambda_2, \qquad \mathrm{Tr}(M_{\mathrm{init}}\, G) \le 1 : \tau.$$

Dual: Multipliers, $\lambda_1 = \lambda_2$, and the Explicit LMI

Associate one multiplier per constraint:

$$f_0 - f_\star + \mathrm{Tr}(M_{0\star} G) \le 0 : \lambda_1, \qquad f_\star - f_0 + \mathrm{Tr}(M_{\star 0} G) \le 0 : \lambda_2, \qquad \mathrm{Tr}(M_{\mathrm{init}}\, G) \le 1 : \tau.$$

Dual equality on F (from the Lagrangian derivation: $\delta_F = 0$):

$$\underbrace{v_{\mathrm{perf}}}_{=0} - \tau \underbrace{v_{\mathrm{init}}}_{=0} - \lambda_1 v_{0\star} - \lambda_2 v_{\star 0} = 0$$

$$-\lambda_1 \begin{pmatrix} 1 \\ -1 \end{pmatrix} - \lambda_2 \begin{pmatrix} -1 \\ 1 \end{pmatrix} = (\lambda_2 - \lambda_1) \begin{pmatrix} 1 \\ -1 \end{pmatrix} = 0 \quad \Rightarrow \quad \lambda_1 = \lambda_2 =: \lambda.$$

Dual: Multipliers, $\lambda_1 = \lambda_2$, and the Explicit LMI

Associate one multiplier per constraint:

$$f_0 - f_\star + \mathrm{Tr}(M_{0\star} G) \le 0 : \ \lambda_1, \qquad f_\star - f_0 + \mathrm{Tr}(M_{\star 0} G) \le 0 : \ \lambda_2, \qquad \mathrm{Tr}(M_{\mathrm{init}}\, G) \le 1 : \ \tau.$$

Dual equality on F (from the Lagrangian derivation: $\delta_F = 0$):

$$\underbrace{v_{\mathrm{perf}}}_{=0} - \tau \underbrace{v_{\mathrm{init}}}_{=0} - \lambda_1 v_{0\star} - \lambda_2 v_{\star 0} = 0$$

$$-\lambda_1 \begin{pmatrix} 1 \\ -1 \end{pmatrix} - \lambda_2 \begin{pmatrix} -1 \\ 1 \end{pmatrix} = (\lambda_2 - \lambda_1) \begin{pmatrix} 1 \\ -1 \end{pmatrix} = 0 \quad \Rightarrow \quad \lambda_1 = \lambda_2 =: \lambda.$$

Dual LMI on G: $S := M_{\mathrm{perf}} - \tau\, M_{\mathrm{init}} - \lambda(M_{0\star} + M_{\star 0}) \preccurlyeq 0, \quad \tau, \lambda \ge 0.$

Dual: Multipliers, $\lambda_1 = \lambda_2$, and the Explicit LMI

Associate one multiplier per constraint:

$$f_0 - f_\star + \mathrm{Tr}(M_{0\star} G) \le 0 : \ \lambda_1, \qquad f_\star - f_0 + \mathrm{Tr}(M_{\star 0} G) \le 0 : \ \lambda_2, \qquad \mathrm{Tr}(M_{\mathrm{init}}\, G) \le 1 : \ \tau.$$

Dual equality on F (from the Lagrangian derivation: $\delta_F = 0$):

$$\underbrace{v_{\mathrm{perf}}}_{=0} - \tau \ \underbrace{v_{\mathrm{init}}}_{=0} - \lambda_1 \, v_{0\star} - \lambda_2 \, v_{\star 0} = 0$$

$$-\lambda_1 \begin{pmatrix} 1 \\ -1 \end{pmatrix} - \lambda_2 \begin{pmatrix} -1 \\ 1 \end{pmatrix} = (\lambda_2 - \lambda_1) \begin{pmatrix} 1 \\ -1 \end{pmatrix} = 0 \quad \Rightarrow \quad \lambda_1 = \lambda_2 =: \lambda.$$

Dual LMI on G: $S := M_{\mathrm{perf}} - \tau \, M_{\mathrm{init}} - \lambda(M_{0\star} + M_{\star 0}) \preccurlyeq 0, \quad \tau, \lambda \ge 0.$

Explicit 2×2, using $M_{0\star} + M_{\star 0} = \frac{1}{2(L-\mu)} \begin{pmatrix} 2\mu L & -(L+\mu) \\ -(L+\mu) & 2 \end{pmatrix}$:

$$S = \begin{pmatrix} 1 - \tau - \dfrac{\mu L \lambda}{L-\mu} & -\gamma + \dfrac{(L+\mu)\lambda}{2(L-\mu)} \\ -\gamma + \dfrac{(L+\mu)\lambda}{2(L-\mu)} & \gamma^2 - \dfrac{\lambda}{L-\mu} \end{pmatrix} \preccurlyeq 0.$$

$\rightarrow$ analytical solution in the Lab

Summary: Three Equivalent Formulations

Naive

$$\tau^* = \sup_{\substack{f \in \mathcal{F}_{\mu,L} \\ x_0}} \frac{\mathrm{Perf}(x_1)}{\mathrm{Init}(x_0)}$$

with $\mathrm{Init}(x_0) = \|x_0 - x_\star\|^2$,
$\mathrm{Perf}(x_1) = \|x_1 - x_\star\|^2$.

- × Infinite-dim.
- × Non-convex
- × No algorithm

Primal SDP

$$\tau^* = \sup_{\substack{G \succcurlyeq 0,\, F \\ \text{s.t. interp.}}} \mathrm{Tr}(M_{\mathrm{perf}}\, G)$$

- ✓ Convex SDP
- ✓ Finite-dim.
- → Feasible (G^*, F^*) = **counter-example**

Dual SDP

$$\tau^* = \inf_{\substack{\lambda, \tau \geq 0 \\ S \preccurlyeq 0}} \tau$$

- ✓ Convex SDP
- → Feasible τ = **valid rate**
- → Feasible λ = **proof cert.**

Summary: Three Equivalent Formulations

Naive

$$\tau^* = \sup_{\substack{f \in \mathcal{F}_{\mu,L} \\ x_0}} \frac{\mathrm{Perf}(x_1)}{\mathrm{Init}(x_0)}$$

with $\mathrm{Init}(x_0) = \|x_0 - x_\star\|^2$, $\mathrm{Perf}(x_1) = \|x_1 - x_\star\|^2$.

- × Infinite-dim.
- × Non-convex
- × No algorithm

Primal SDP

$$\tau^* = \sup_{\substack{G \succcurlyeq 0,\, F \\ \text{s.t. interp.}}} \mathrm{Tr}(M_{\mathrm{perf}}\, G)$$

- ✓ Convex SDP
- ✓ Finite-dim.
- → Feasible (G^*, F^*) = **counter-example**

Dual SDP

$$\tau^* = \inf_{\substack{\lambda, \tau \geq 0 \\ S \preccurlyeq 0}} \tau$$

- ✓ Convex SDP
- → Feasible τ = **valid rate**
- → Feasible λ = **proof cert.**

Strong duality (Slater): $\tau^*_{\mathrm{primal}} = \tau^*_{\mathrm{dual}}$.

Any dual feasible (τ, λ) with $S \preccurlyeq 0$: **valid rate** τ AND **machine-checkable proof** (weighted combination of interpolation inequalities with weights λ).

Outline

Summary & References

What we covered:

1. GD convergence proof (co-coercivity)
2. PEP = optimization over functions $\rightarrow$ SDP
3. Step-by-step for $\|x_1 - x_\star\|^2$ and $f(x_1) - f_\star$
4. Generic framework (any algo, any class, any metric)
5. Primal = counter-example; Dual = proof certificate
6. Interpolation conditions & homogeneity
7. Minimal PEPit code

Key references:

- Drori & Teboulle (2014). Performance of first-order methods.
- Taylor, Hendrickx, Glineur (2017). Smooth strongly convex interpolation.
- Taylor et al. (2018). Lyapunov functions for first-order methods.
- Goujaud et al. (2022). PEPit: computer-assisted worst-case analyses.
- Github: `https://github.com/PerformanceEstimation/PEPit`
- Docs: `https://pepit.readthedocs.io`

Summary & References

What we covered:

1. GD convergence proof (co-coercivity)
2. PEP = optimization over functions $\rightarrow$ SDP
3. Step-by-step for $\|x_1 - x_\star\|^2$ and $f(x_1) - f_\star$
4. Generic framework (any algo, any class, any metric)
5. Primal = counter-example; Dual = proof certificate
6. Interpolation conditions & homogeneity
7. Minimal PEPit code

Key references:

- Drori & Teboulle (2014). Performance of first-order methods.
- Taylor, Hendrickx, Glineur (2017). Smooth strongly convex interpolation.
- Taylor et al. (2018). Lyapunov functions for first-order methods.
- Goujaud et al. (2022). PEPit: computer-assisted worst-case analyses.
- Github: `https://github.com/PerformanceEstimation/PEPit`
- Docs: `https://pepit.readthedocs.io`

References I

Appendix