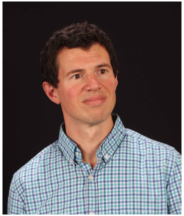
PEPit: a Python package for worst-case analysis of first-order optimization methods and their continuous versions

Céline Moucer

ICCOPT, July 2022

Joint work with

Baptiste Goujaud



Julien Hendrickx

François Glineur

Aymeric Dieuleveut

Adrien Taylor

Francis Bach

Outline

Motivations

- A principled approach to worst-case analysis of optimization methods and their continuous-time limit
- Quick numerical evaluation of (new) first-order methods
- Reproducible research

PEPit:

- **A python package**[1] for worst-scale analyses of a large family of first-order methods (so-called Performance Estimation Problems - PEPs, see PESTO for the matlab version [9])
- Technique based on semidefinite programming (SDP)
- An example: **gradient flow originating from strongly convex functions** [2].

[1] [3] **B. Goujaud et al.** PEPit: computer-assisted worst-case analyses of first-order optimization methods in Python

[2] [5] **M., A. Taylor, and F. Bach.** A systematic approach to Lyapunov analyses of continuous-time models in convex optimization

First-order methods in convex optimization

A very popular setting:

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

where f is convex, differentiable, and $x_\star \in \mathbf{R}^d$ an optimal point.

- **First-order methods**: low-cost per iteration, accuracy is not critical (machine learning, signal processing, etc.)

$$x_{k+1} \in \mathbf{Span}(x_0, \nabla f(x_0), ..., \nabla f(x_{k+1}))$$

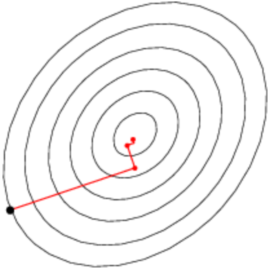


Figure: Convex function and optimization algorithm

First-order methods in convex optimization

Gradient descent with fixed step size $\gamma > 0$:

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

First-order methods in convex optimization

Gradient descent with fixed step size $\gamma > 0$:

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

- **Ordinary differential equations (ODEs)**: When taking the step size γ to 0, it is directly related to the gradient flow,

$$\dot{X}_t = -\nabla f(X_t), \; X_0 = x_0 \in \mathbf{R}^d,$$

where X_t verifies $X_{t_k} \approx x_k$ with the identification $t_k = \gamma k$.

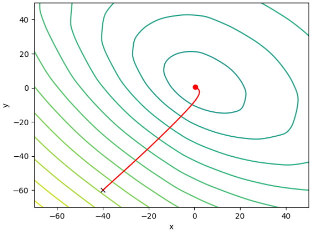


Figure: Integration of the gradient flow for a logistic regression problem.

Optimization methods and ODEs: convergence guarantees

- **First-order methods**: given a class of functions $\mathcal{F}$, a starting point $x_0 \in \mathbf{R}^d$, and given gradient descent with step size $\gamma > 0$

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

the goal is to quantify the convergence speed to an optimum $x_\star$ in a small number of steps k,

$$\|x_k - x_\star\|^2 \leqslant \tau(k, \mathcal{F}, \gamma)\|x_0 - x_\star\|^2.$$

Optimization methods and ODEs: convergence guarantees

- **First-order methods**: given a class of functions $\mathcal{F}$, a starting point $x_0 \in \mathbf{R}^d$, and given gradient descent with step size $\gamma > 0$

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

the goal is to quantify the convergence speed to an optimum $x_\star$ in a small number of steps k,

$$\|x_k - x_\star\|^2 \leqslant \tau(k, \mathcal{F}, \gamma) \|x_0 - x_\star\|^2.$$

- **ODEs** : given a class of function $\mathcal{F}$, a starting point $x_0 \in \mathbf{R}^d$, the gradient flow starting is given by,

$$\frac{d}{dt} X_t = -\nabla f(X_t),$$

the goal is to quantify the convergence speed to an $x_\star$,

$$\|X_t - x_\star\|^2 \leqslant \tau(t, \mathcal{F}) \|x_0 - x_\star\|^2.$$

Optimization methods and ODEs: convergence guarantees

- **First-order methods**: given a class of functions $\mathcal{F}$, a starting point $x_0 \in \mathbf{R}^d$, and given gradient descent with step size $\gamma > 0$

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

the goal is to quantify the convergence speed to an optimum $x_\star$ in a small number of steps k,

$$\|\nabla f(x_k)\|^2 \leqslant \tilde{\tau}(k, \mathcal{F}, \gamma)\|x_0 - x_\star\|^2.$$

- **ODEs** : given a class of function $\mathcal{F}$, a starting point $x_0 \in \mathbf{R}^d$, the gradient flow starting is given by,

$$\frac{d}{dt}X_t = -\nabla f(X_t),$$

the goal is to quantify the convergence speed to an $x_\star$,

$$f(X_t) - f_\star \leqslant \tilde{\tau}(t, \mathcal{F})\,(f(x_0) - f(x_\star)).$$

Convex optimization setting

Common assumptions:

- f is convex and differentiable,
- A differentiable function f is **L-smooth** if and only if it satisfies

$$\|\nabla f(x) - \nabla f(y)\| \leqslant L\|x - y\|.$$

- A convex differentiable function f is **μ-strongly convex** if and only if it satisfies

$$\|\nabla f(x) - \nabla f(y)\| \geqslant \mu\|x - y\|.$$

$\mathcal{F}_{\mu,L}$ is the family of a L-smooth μ-strongly convex functions, with $0 \leq \mu \leq L \leq +\infty$.

Performance estimation problems (PEPs)

Main ideas:

1. Optimization methods and associated ODEs are usually studied via worst-case analyses.
2. Convergence proofs are combinations of inequalities (from methods and problem class).
3. Automated search for combinations of inequalities.

Performance estimation problems (PEPs)

Main ideas:

1. Optimization methods and associated ODEs are usually studied via worst-case analyses.
2. Convergence proofs are combinations of inequalities (from methods and problem class).
3. Automated search for combinations of inequalities.

References:

- Initiated by Drori and Teboulle (2012) [2]
- Analyses of first-order methods and design of proofs by Taylor et al. (2017) [8]

An example: the gradient flow

We consider the **gradient flow** starting from $x_0 \in \mathbf{R}^d$, and originating from differentiable functions f:

$$\frac{d}{dt} X_t = -\nabla f(X_t).$$

An example: the gradient flow

We consider the **gradient flow** starting from $x_0 \in \mathbf{R}^d$, and originating from differentiable functions f:

$$\frac{d}{dt} X_t = -\nabla f(X_t).$$

Lyapunov functions: given a trajectory X_t, many proofs construct a Lyapunov function $\mathcal{V} : x, t \in \mathbf{R}^d, \mathbf{R}^+ \to \mathbf{R}$, such that,

1. $\mathcal{V}(x, t) = 0 \iff x = x_\star$,
2. $\mathcal{V}(X_t, t) \geqslant 0$,
3. $\frac{d}{dt}\mathcal{V}(X_t, t) \leqslant 0$.

An example: the gradient flow

We consider the **gradient flow** starting from $x_0 \in \mathbf{R}^d$, and originating from differentiable functions f:

$$\frac{d}{dt} X_t = -\nabla f(X_t).$$

Lyapunov functions: given a trajectory X_t, many proofs construct a Lyapunov function $\mathcal{V} : x, t \in \mathbf{R}^d, \mathbf{R}^+ \to \mathbf{R}$, such that,

1. $\mathcal{V}(x, t) = 0 \iff x = x_\star$,
2. $\mathcal{V}(X_t, t) \geqslant 0$,
3. $\frac{d}{dt}\mathcal{V}(X_t, t) \leqslant 0$.

For example, let us consider the function $\mathcal{V}(X_t, t) = f(X_t)$:

$$\frac{d}{dt}\mathcal{V}(X_t, t) = \dot{X_t}^T \nabla f(X_t) = -\|\nabla f(X_t))\|^2 \leqslant 0.$$

Worst-case guarantee using Lyapunov functions

We consider the **gradient flow** starting from $x_0 \in \mathbf{R}^d$, and originating from strongly convex functions $f \in \mathcal{F}_{\mu,\infty}$:

$$\frac{d}{dt}X_t = -\nabla f(X_t).$$

Worst-case guarantee: given a Lyapunov function $\mathcal{V}$, we look for (the largest) values $\tau(\mu) \geqslant 0$ such that

$$\frac{d}{dt}\mathcal{V}(X_t) \leqslant -\tau(\mu)\mathcal{V}(X_t),$$

is true for all functions $f \in \mathcal{F}_{\mu,\infty}$, and all trajectories X_t.

Worst-case guarantee using Lyapunov functions

We consider the **gradient flow** starting from $x_0 \in \mathbf{R}^d$, and originating from strongly convex functions $f \in \mathcal{F}_{\mu,\infty}$:

$$\frac{d}{dt}X_t = -\nabla f(X_t).$$

Worst-case guarantee: given a Lyapunov function $\mathcal{V}$, we look for (the largest) values $\tau(\mu) \geqslant 0$ such that

$$\frac{d}{dt}\mathcal{V}(X_t) \leqslant -\tau(\mu)\mathcal{V}(X_t),$$

is true for all functions $f \in \mathcal{F}_{\mu,\infty}$, and all trajectories X_t.

Integrating between 0 and t: $\mathcal{V}(X_t) \leqslant e^{-\tau(\mu)t}\mathcal{V}(x_0)$.

Worst-case guarantee using Lyapunov functions

We consider the **gradient flow** starting from $x_0 \in \mathbf{R}^d$, and originating from strongly convex functions $f \in \mathcal{F}_{\mu,\infty}$:

$$\frac{d}{dt} X_t = -\nabla f(X_t).$$

Worst-case guarantee: given a Lyapunov function $\mathcal{V}$, we look for (the largest) values $\tau(\mu) \geqslant 0$ such that

$$\frac{d}{dt}\mathcal{V}(X_t) \leqslant -\tau(\mu)\mathcal{V}(X_t),$$

is true for all functions $f \in \mathcal{F}_{\mu,\infty}$, and all trajectories X_t.

Integrating between 0 and t: $\mathcal{V}(X_t) \leqslant e^{-\tau(\mu)t}\mathcal{V}(x_0)$.

Reformulation as an optimization problem:

$$\begin{aligned} -\tau(\mu) = \max_{X_t \in \mathbf{R}^d,\ f \in \mathcal{F}_{\mu,\infty}} &\ \frac{d}{dt}\mathcal{V}(X_t), \\ \text{subject to } &\ \mathcal{V}(X_t) = 1, \\ &\ \dot{X}_t = -\nabla f(X_t). \end{aligned}$$

Worst-case guarantee using Lyapunov functions

We consider the **gradient flow** starting from $x_0 \in \mathbf{R}^d$, and originating from strongly convex functions $f \in \mathcal{F}_{\mu,\infty}$:

$$\frac{d}{dt} X_t = -\nabla f(X_t).$$

Given the Lyapunov function $\mathcal{V}(X_t) = f(X_t) - f_\star$,

$$\begin{aligned} -\tau(\mu) = \max_{X_t,\ f \in \mathcal{F}_{\mu,\infty}} \ & \dot{X}_t^T \nabla f(X_t), \\ \text{subject to } & f(X_t) - f_\star = 1, \\ & \dot{X}_t = -\nabla f(X_t). \end{aligned}$$

Worst-case guarantee using Lyapunov functions

We consider the **gradient flow** starting from $x_0 \in \mathbf{R}^d$, and originating from strongly convex functions $f \in \mathcal{F}_{\mu,\infty}$:

$$\frac{d}{dt} X_t = -\nabla f(X_t).$$

Given the Lyapunov function $\mathcal{V}(X_t) = f(X_t) - f_\star$,

$$\begin{aligned} -\tau(\mu) = \max_{X_t,\ f \in \mathcal{F}_{\mu,\infty}} \ & \dot{X}_t{}^T \nabla f(X_t), \\ \text{subject to } & f(X_t) - f_\star = 1, \\ & \dot{X}_t = -\nabla f(X_t). \end{aligned}$$

This infinite dimensional problem can be reformulated as an SDP.

A reformulation using sampling

Reformulation as a feasibility condition over the class of functions,

$$\begin{aligned} \max_{(X_i, g_i, f_i)_{t,\star}} \quad & -\|g_t\|^2, \\ \text{subject to } & f_t - f_\star = 1, \\ & \exists f \in \mathcal{F}_{\mu,\infty} : \begin{cases} g_t = \nabla f(X_t) & f_t = f(X_t), \\ g_\star = \nabla f(x_\star) = 0 & f_\star = f(x_\star). \end{cases} \end{aligned}$$

A reformulation using sampling

Reformulation as a feasibility condition over the class of functions,

$$\max_{(X_i, g_i, f_i)_{t,\star}} \; -\|g_t\|^2,$$

$$\text{subject to } f_t - f_\star = 1,$$

$$\exists f \in \mathcal{F}_{\mu,\infty} : \begin{cases} g_t = \nabla f(X_t) & f_t = f(X_t), \\ g_\star = \nabla f(x_\star) = 0 & f_\star = f(x_\star). \end{cases}$$

Theorem [8, Theorem 4]

Set $\{(X_i, g_i, f_i)\}_{i \in I}$ if $\mathcal{F}_{\mu,\infty}$ interpolable if and only if the following set of conditions holds for every pair of indices $i \in I$ and $j \in J$

$$f_i - f_j - g_j^T (X_i - X_j) \geqslant \frac{\mu}{2} \|X_i - X_j\|^2.$$

A reformulation as a finite-dimensional quadratic problem

Reformulation as a nonconvex QCQP (quadratically constrained quadratic program):

$$\begin{aligned} \max_{(X_i,g_i,f_i)_{t,\star},\, g_\star=0} \; & -\|g_t\|^2, \\ \text{subject to } & (f_t - f_\star) = 1, \\ & f_i - f_j - g_j^T(X_i - X_j) \geqslant \frac{\mu}{2}\|X_i - X_j\|^2, \; \forall i,j = t,\star. \end{aligned}$$

Linear in $(f_t, f_\star)$ and quadratic in $(X_t, X_\star, g_t)$.

A reformulation as a finite-dimensional quadratic problem

Reformulation as a nonconvex QCQP (quadratically constrained quadratic program):

$$\begin{aligned} \max_{(X_i, g_i, f_i)_{t,\star}, g_\star = 0} \; & -\|g_t\|^2, \\ \text{subject to } & (f_t - f_\star) = 1, \\ & f_i - f_j - g_j^T(X_i - X_j) \geqslant \frac{\mu}{2}\|X_i - X_j\|^2, \;\; \forall i, j = t, \star. \end{aligned}$$

Linear in $(f_t, f_\star)$ and quadratic in $(X_t, X_\star, g_t)$.

A finite-dimensional problem that is still nonconvex.

A reformulation into an SDP

Let us introduce:

- G be a Gram matrix defined by $G = \begin{pmatrix} \|X_t - x_\star\|^2 & \langle X_t - x_\star, g_t\rangle \\ \langle X_t - x_\star, g_t\rangle & \|g_t\|^2 \end{pmatrix} \succeq 0$
- the vector $F = f_t - f_\star$

Formulation into an SDP,

$$\begin{aligned} \max_{G \succeq 0, F} \; & \mathrm{Tr}(A_0 G), \\ \text{subject to } & b_0^T F = 1, \\ & b_1^T F + \mathrm{Tr}(A_1 G) \geqslant 0, \\ & b_2^T F + \mathrm{Tr}(A_2 G) \geqslant 0, \end{aligned}$$

where $A_0 = \begin{pmatrix} 0 & 0 \\ 0 & -1 \end{pmatrix}$, $A_1 = \begin{pmatrix} -\mu/2 & 1/2 \\ 1/2 & 0 \end{pmatrix}$, $A_2 = \begin{pmatrix} -\mu/2 & 0 \\ 0 & 0 \end{pmatrix}$, $b_1 = -1$ and $b_2 = b_0 = 1$.

Linear SDP $\rightarrow$ can be solved numerically.

Automated SDP modeling using PEPit

Goals of the toolbox:

- Performing numerical worst-case analyses
- Avoiding the SDP modeling part
- Avoiding some potential mistakes in the SDP formulation / numerous interpolation inequalities involved
- Describing the method / ODE as the user would have implemented it

See this colab notebook.

PEPit VS known bounds: gradient flow

A closed-form upper bound in the worst-case is given by the lemma.

Lemma

Let f be a μ-strongly convex function, $x_0 \in \mathbf{R}^d$, and $x_\star$ the minimizer of f. The solution X_t to the gradient flow verifies

$$\frac{d}{dt}\left(f(X_t) - f(x_\star)\right) \leqslant -2\mu\left(f(X_t) - f(x_\star)\right),$$

and after integrating between 0 *and* t*,* $f(X_t) - f(x_\star) \leqslant e^{-2\mu t}(f(x_0) - f(x_\star))$.

Proof: Let us define $\mathcal{V}(X_t) = f(X_t) - f_\star$. Then, deriving and using strong convexity, $\frac{d}{dt}\mathcal{V}(X_t) = \dot{X_t}^T \nabla f(X_t) = -\|\nabla f(X_t)\|^2 \leqslant -2\mu(f(X_t) - f_\star) \leqslant -2\mu\mathcal{V}(X_t)$.

PEPit VS known upper bound: gradient flow

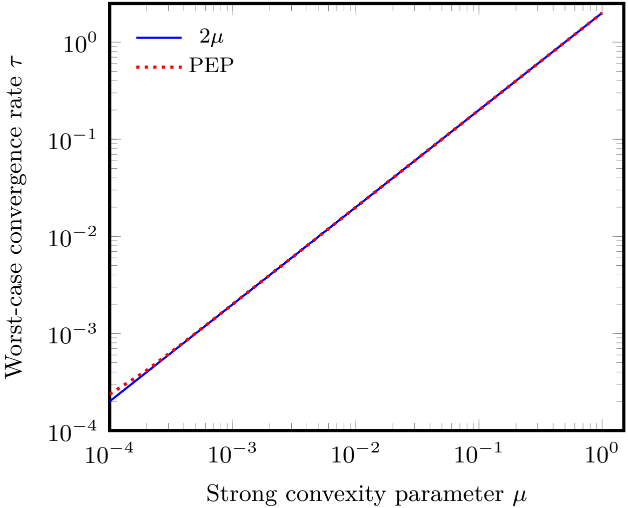


Figure: Worst-case rate τ on the class of quadratic Lyapunov functions for the gradient flow [5].

Important ingredients for the SDP reformulations

Given the gradient flow $\frac{d}{dt}X_t = -\nabla f(X_t)$ originating from strongly convex functions f, we look for the convergence guarantee,

$$\frac{d}{dt}\mathcal{V}(X_t) \leqslant -\tau(\mu)\mathcal{V}(X_t).$$

4 main ingredients:

- **A class of functions** with interpolating conditions linear in G and F

  $\to \mathcal{F}_{\mu,\infty}$
- **An ODE / first-order method**: linear in G and F

  $\to$ *Gradient flow*
- **A performance measure**: linear in G and F

  $\to$ *Derivative* $\frac{d}{dt}\mathcal{V}(X_t)$ *of Lyapunov functions given by* $\mathcal{V}(X_t) = f(X_t) - f_\star$
- **An initial condition**: linear in G and F

  $\to$ *Lyapunov function* $\mathcal{V}(X_t) = 1$

Important ingredients for the SDP reformulations

Given $f \in \mathcal{F}_{\mu,L}$, and an accelerated gradient method starting from $x_0 \in \mathbf{R}^d$, with $\kappa = \frac{\mu}{L}$

$$y_{n+1} = x_{n+1} - \frac{1-\sqrt{\kappa}}{1+\sqrt{\kappa}}(x_{n+1} - x_n), \; x_{n+1} = y_n - \frac{1}{L}\nabla f(x_n),$$

An upper bound guarantee is given in [1],

$$f(x_k) - f_\star \leqslant \left(1-\sqrt{\kappa}\right)^k \left(f(x_0) - f_\star + \frac{\mu}{2}\|x_0 - x_\star\|^2\right).$$

4 main ingredients:

- **A class of functions** with interpolating conditions linear in G and F

  $\rightarrow \mathcal{F}_{\mu,L}$
- **An ODE / first-order method**: linear in G and F

  $\rightarrow$ *An accelerated gradient descent*
- **A performance measure**: linear in G and F

  $\rightarrow f(x_k) - f_\star$
- **An initial condition**: linear in G and F

  $\rightarrow f(x_0) - f_\star + \frac{\mu}{2}\|x_0 - x_\star\|^2$

PEPit VS known upper bound : an accelerated gradient method

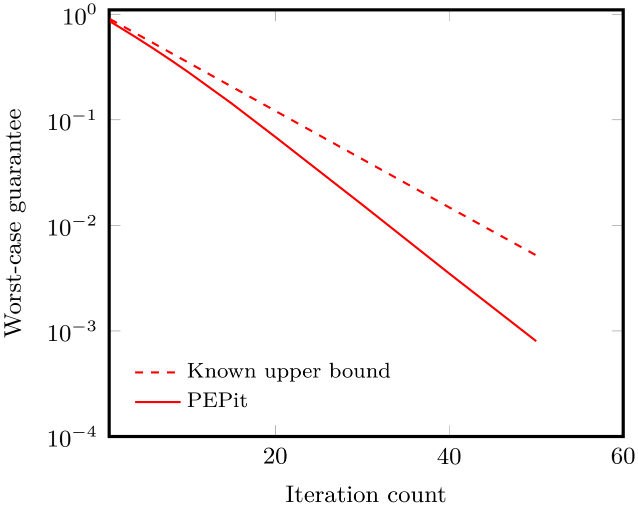


Figure: Accelerated gradient method (with constant momentum): strong convexity parameter fixed to $\mu = 0.1$. Worst-case guarantee on $f(x_n) - f_\star$ as a function of n.

Overview and contents of the toolbox

- **Black-box oracles** include (sub)-gradient steps, proximal steps, linear optimization steps, and their approximate / inexact and Bregman versions.
- **Problem classes** include convex functions (with Lipschitz functions, bounded domains, smoothness, strong-convexity), convex indicator functions, smooth nonconvex functions, monotone operators.
- **Performance measure and initial conditions** include everything that has a convex representation in terms of a Gram matrix:
  - linear in function values
  - quadratic in gradient / iterates

Already about 50 examples in the toolbox:
`https://pepit.readthedocs.io/en/latest/examples.html`

Concluding remarks

Future work:

- Automated export of numerical proofs
- Implementing the automated search for Lyapunov functions [7, 6, 4]
- Additional oracles / class of functions

Concluding remarks

Future work:

- Automated export of numerical proofs
- Implementing the automated search for Lyapunov functions [7, 6, 4]
- Additional oracles / class of functions

Conclusion:

- A tool for validating proofs and worst-case guarantees
- Flexible criterion, initial conditions, oracles, methods, ...
- Easy to implement for the user
- Intuition on methods behaviors
- Possibly a great help in the review process! :)

PEPit: a Python package for worst-case analysis of first-order optimization methods and their continuous versions

Céline Moucer

ICCOPT, July 2022

Thanks!
Any question?

Dual formulation

- A numerical proof assistant:

  Let us recall the PEP for gradient flow in its SDP reformulation

$$
\begin{aligned}
\max_{G \succeq 0, F \in \mathbf{R}^2} \quad & \mathrm{Tr}(A_0 G), \\
\text{subject to } \quad & b_0^T F = 1, \; (\tau) \\
& b_1^T F + \mathrm{Tr}(A_1 G) \geqslant 0, \; (\lambda_1 \geqslant 0) \\
& b_2^T F + \mathrm{Tr}(A_2 G) \geqslant 0, \; (\lambda_2 \geqslant 0)
\end{aligned}
$$

Dual formulation

- A numerical proof assistant:

  Let us recall the PEP for gradient flow in its SDP reformulation

$$
\begin{aligned}
\max_{G \succeq 0, F \in \mathbf{R}^2} \quad & \operatorname{Tr}(A_0 G), \\
\text{subject to } \quad & b_0^T F = 1, \; (\tau) \\
& b_1^T F + \operatorname{Tr}(A_1 G) \geqslant 0, \; (\lambda_1 \geqslant 0) \\
& b_2^T F + \operatorname{Tr}(A_2 G) \geqslant 0, \; (\lambda_2 \geqslant 0)
\end{aligned}
$$

  Thanks to strong duality (Slater point), it is possible to select the inequalities (dual variables) involves at the optimum,

$$
\begin{aligned}
\min_{\tau} \quad & \tau, \\
& \lambda_1 b_1 + \lambda_2 b_1 - \tau b_0 \leqslant 0, \\
& \lambda_1 A_1 + \lambda_2 A_2 + A_0 \preceq 0, \\
& \lambda_1, \lambda_2 \geqslant 0.
\end{aligned}
\tag{1}
$$

Optimizing over classes of Lyapunov functions

- A numerical proof assistant
- Searching for Lyapunov functions :

For the gradient flow, what is the best choice of Lyapunov function minimizing worst-case convergence guarantee?

$$\mathcal{V}_{a,c}(X_t) = a(f(X_t) - f_\star) + c\|X_t - x_\star\|^2.$$

Then the maximization problem for the gradient flow becomes,

$$\begin{aligned} -\tau = \min_{a,c\geqslant 0} \max_{X_t\in\mathbf{R}^d,\ f\in\mathcal{F}_{\mu,\infty}} & \frac{d}{dt}\mathcal{V}(X_t), \\ \text{subject to } & \mathcal{V}(X_t) = 1, \\ & \dot{X}_t = -\nabla f(X_t). \end{aligned} \tag{2}$$

References I

[1] Alexandre d'Aspremont, Damien Scieur, and Adrien Taylor. "Acceleration Methods". In: *Foundations and Trends® in Optimization* 5.1-2 (2021), pp. 1–245.

[2] Yoel Drori and Marc Teboulle. "Performance of first-order methods for smooth convex minimization: a novel approach". In: *Mathematical Programming* 145.1 (2014), pp. 451–482.

[3] Baptiste Goujaud et al. *PEPit: computer-assisted worst-case analyses of first-order optimization methods in Python*. 2022. DOI: 10.48550/ARXIV.2201.04040. URL: https://arxiv.org/abs/2201.04040.

[4] Laurent Lessard, Benjamin Recht, and Andrew Packard. "Analysis and design of optimization algorithms via integral quadratic constraints". In: *SIAM Journal on Optimization* 26.1 (2016), pp. 57–95.

[5] Céline Moucer, Adrien Taylor, and Francis Bach. *A systematic approach to Lyapunov analyses of continuous-time models in convex optimization*. 2022.

References II

[6] Adrien Taylor and Francis Bach. *Stochastic first-order methods: non-asymptotic and computer-aided analyses via potential functions.* 2021. arXiv: 1902.00947 [math.OC].

[7] Adrien Taylor, Bryan Van Scoy, and Laurent Lessard. *Lyapunov Functions for First-Order Methods: Tight Automated Convergence Guarantees.* 2018. arXiv: 1803.06073 [math.OC].

[8] Adrien B Taylor, Julien M Hendrickx, and François Glineur. "Smooth strongly convex interpolation and exact worst-case performance of first-order methods". In: *Mathematical Programming* 161.1 (2017), pp. 307–345.

[9] Adrien B. Taylor, Julien M. Hendrickx, and François Glineur. "Performance estimation toolbox (PESTO): Automated worst-case analysis of first-order optimization methods". In: *2017 IEEE 56th Annual Conference on Decision and Control (CDC)*. 2017, pp. 1278–1283. DOI: 10.1109/CDC.2017.8263832.