AutoLyap: A Python package for computer-assisted Lyapunov analyses for first-order methods

Manu Upadhyaya[⋆] Adrien B. Taylor[†] Sebastian Banert[‡] Pontus Giselsson[⋆]

[⋆]Department of Automatic Control, Lund University, Lund, Sweden
{manu.upadhyaya, pontus.giselsson}@control.lth.se

[†]INRIA & D.I. École Normale Supérieure, CNRS & PSL Research University, Paris, France
adrien.taylor@inria.fr

[‡]Center for Industrial Mathematics, University of Bremen, Bremen, Germany
banert@uni-bremen.de

Abstract

We introduce AutoLyap, a Python package designed to automate Lyapunov analyses for a wide class of first-order methods for solving structured optimization and inclusion problems. Lyapunov analyses are structured proof patterns commonly used to establish convergence results for first-order methods. Building on previous work [2], the core idea behind AutoLyap is to recast the verification of the existence of a Lyapunov analysis as a semidefinite programming (SDP) problem, which can then be solved numerically using standard SDP solvers. Users of the package specify (i) the optimization or inclusion problem, (ii) the first-order method in question, and (iii) the type of Lyapunov analysis they wish to verify. Once these inputs are provided, AutoLyap handles the SDP modeling and proceeds with the numerical solution of the SDP. We numerically verify—and in some cases extend—numerous established convergence results, demonstrating the practical relevance of our approach.

Keywords. Software, first-order methods, operator splitting methods, performance estimation, Lyapunov analysis, semidefinite programming

1 Introduction

Optimization and fixed-point algorithms are fundamental tools in various fields, such as automatic control, machine learning, and inverse problems. First-order methods, in particular, are favored for their simplicity and scalability when handling large-scale problems. However, the theoretical justification and analysis of first-order methods are typically reserved for experts in the field. The primary objective of this work is to automate the most challenging aspects of the analysis process by generating convergence proofs in a straightforward and reproducible manner.

1.1 Scope

This extended abstract highlights our methodology through a simple example. For the complete package documentation, theoretical development, full proofs, and a broader set of examples, readers are referred to the full-length version.

2 Nesterov's fast gradient method

Consider the optimization problem

$$\underset{x\in\mathbb{R}^n}{\operatorname{minimize}}\ f(x) \tag{1}$$

where $f:\mathbb{R}^n\to\mathbb{R}$ is convex with L-Lipschitz continuous gradient for some constant $L>0$. In the seminal paper [1], Nesterov presents a fast gradient method that achieves the optimal convergence rate (up to a constant) among all first-order methods that solve (1). It is given by initializing $\lambda_0 = 1$ and $x^{-1}, x^0 \in \mathbb{R}^n$, and letting

$$(\forall k \in \mathbb{N}_0) \quad \left[\begin{aligned} &y^k = x^k + \alpha_k(x^k - x^{k-1}),\\ &x^{k+1} = y^k - \tfrac{1}{L}\nabla f(y^k), \end{aligned}\right. \quad \text{where} \quad \left[\begin{aligned} &\alpha_k = \tfrac{\lambda_k - 1}{\lambda_{k+1}},\\ &\lambda_{k+1} = \tfrac{1+\sqrt{1+4\lambda_k^2}}{2}. \end{aligned}\right. \tag{2}$$

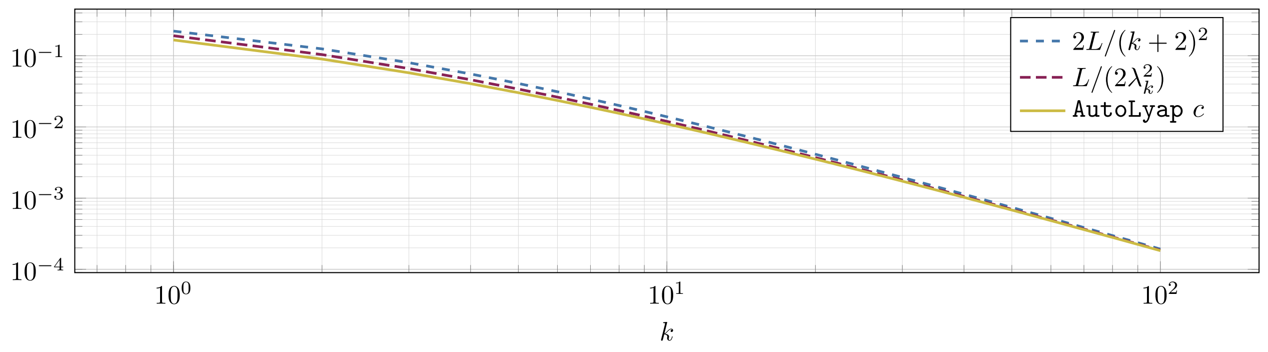

Figure 1: Convergence rates for (2), with `AutoLyap` tightening the classical rates of [1]

Based on a Lyapunov analysis, [1] gives the convergence rates

$$(\forall k \in \mathbb{N})(\forall x^\star \in \operatorname*{argmin}_{x\in\mathbb{R}^n} f(x)) \quad f(x^k) - f(x^\star) \leq \frac{L\|x^0 - x^\star\|^2}{2\lambda_k^2} \leq \frac{2L\|x^0 - x^\star\|^2}{(k+2)^2}. \tag{3}$$

The fundamental proof structure behind obtaining the rates in (3) is by constructing a sequence of chained Lyapunov functions $\mathcal{V}(0), \ldots, \mathcal{V}(k)$ such that

$$\mathcal{V}(k) \leq \mathcal{V}(k-1) \leq \ldots \leq \mathcal{V}(1) \leq c\mathcal{V}(0), \tag{4}$$

under quadratic ansatzes[1]

$$\begin{aligned}(\forall \tau \in [\![0,k]\!]) \quad \mathcal{V}(\tau) = &\langle (x^\tau, x^{\tau-1}, x^\star, \nabla f(x^\tau), \nabla f(y^\tau)), (Q_\tau \otimes I)(x^\tau, x^{\tau-1}, x^\star, \nabla f(x^\tau), \nabla f(y^\tau))\rangle \\ &+ \langle q_\tau, (f(x^\tau), f(y^\tau), f(x^\star))\rangle,\end{aligned}$$

for parameters $Q_\tau^\top = Q_\tau \in \mathbb{R}^{5\times5}$ and $q_\tau \in \mathbb{R}^3$, where the initial parameters (Q_0, q_0) are chosen such that $\mathcal{V}(0) = \|x^0 - x^\star\|^2$ and the final parameters (Q_k, q_k) are chosen such that $\mathcal{V}(k) = f(x^k) - f(x^\star)$, and the goal is to minimize the constant $c \geq 0$.

The key idea behind the `AutoLyap` package (specialized to this particular method and Lyapunov analysis) is that minimizing c under the constraint that (4) holds can equivalently be written as an SDP and solved numerically. This is done below for the case of $k = 10$. The same procedure is repeated while sweeping k from 1 to 100, and the result is presented in Figure 1.

```python
from autolyap.problemclass import InclusionProblem, SmoothConvex
from autolyap.algorithms import NesterovFastGradientMethod
from autolyap.automated_analysis_tools import IterationDependent

k = 10 # Iteration budget
L = 1 # The Lipschitz constant
components_list = [
    SmoothConvex(L=L), # The properties of the function f
]
problem = InclusionProblem(components_list) # Considers 0 ∈ {∇f(x)}, i.e., ∇f(x) = 0
algorithm = NesterovFastGradientMethod(L=L) # Contains the update equation (2)

(Q_0, q_0) = IterationDependent.get_parameters_distance_to_solution(algorithm, k=0) #
    Initial Lyapunov function parameters

(Q_k, q_k) = IterationDependent.get_parameters_function_value_suboptimality(algorithm,
    k=k, j=2) # Final Lyapunov function parameters

c = IterationDependent.verify_iteration_dependent_Lyapunov(problem, algorithm, k, Q_0,
    Q_k, q_0, q_K) # Formulates and solves the SDP
```

References

[1] Y. Nesterov. “A method for solving the convex programming problem with convergence rate $\mathcal{O}(1/k^2)$”. In: *Dokl. Akad. Nauk SSSR* 269.3 (1983), pp. 543–547.

[2] M. Upadhyaya, S. Banert, A. B. Taylor, and P. Giselsson. “Automated tight Lyapunov analysis for first-order methods”. In: *Mathematical Programming* 209 (2025), pp. 133–170. DOI: 10.1007/s10107-024-02061-8.

[1] Here $\otimes$ denotes the Kronecker product, and $\langle\cdot,\cdot\rangle$ the standard dot product.