

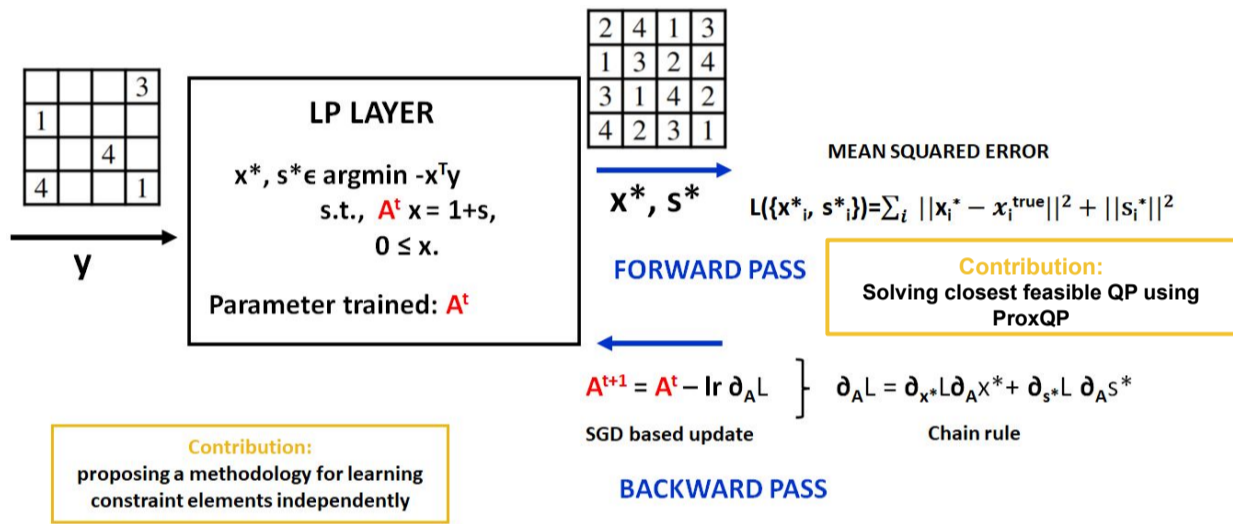


## Leveraging augmented-Lagrangian techniques for differentiating over infeasible quadratic programs in machine learning

A. Bambade<sup>1,2,3</sup> F. Schramm<sup>1,2</sup> A. Taylor<sup>1,2</sup> J. Carpentier<sup>1,2</sup>

<sup>1</sup>Inria Paris, France <sup>2</sup>Département d'informatique de l'ENS, PSL Research University, Paris, France <sup>3</sup>École des Ponts, Marne-la-Vallée, France

### Motivation and solution pipeline



A. Chiche, J.-C. Gilbert (2016)

### Augmented Lagrangian methods in practice with ProxSuite library

# ProxSuite

THE ADVANCED PROXIMAL OPTIMIZATION TOOLBOX

License: [BSD-2-Clause](#) docs: [pdf](#) [video](#) Try CI: [Linux/OS/Windows](#) - [Conda](#) [pip](#) [pypi package](#) [9.6.1](#) [Anaconda.org](#) [0.6.1](#)

- ✓ **fast**: C++ implementation, with homemade linear Cholesky solver
- ✓ **scalable**: various backends for dense, sparse and matrix-free optimization
- ✓ **easy-to-use**: API closed to OSQP, Python and Julia bindings
- ✓ **open-source**: BSD-license, easily installable

| Conda | Files | Labels | Badges |
|-------|-------|--------|--------|
|-------|-------|--------|--------|

📄 License: BSD-2-Clause  
🏠 Home: <https://github.com/simple-robotics/proxsuite>  
</> Development: <https://github.com/simple-robotics/proxsuite>  
📦 160860 total downloads  
📅 Last upload: 1 month and 25 days ago

### Summary

PyPI link  
<https://pypi.org/project/proxsuite>

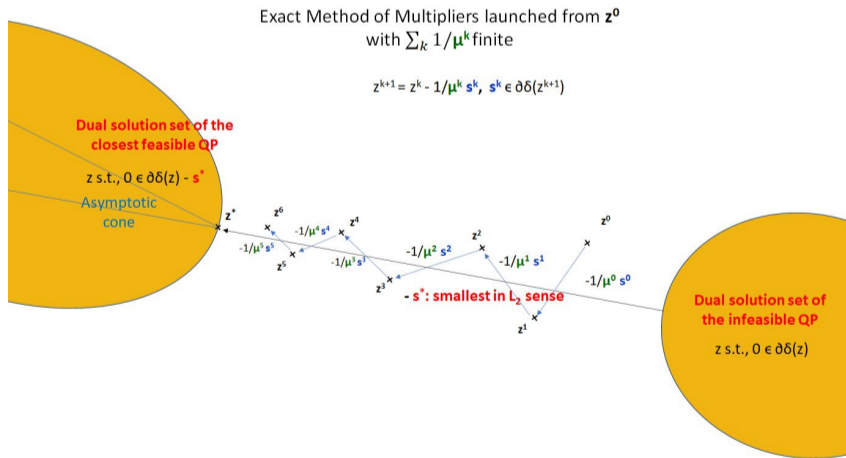
Total downloads  
180,196

Total downloads - 30 days  
11,791

Total downloads - 7 days  
3,257

**Software contribution**

### The forward pass: Augmented Lagrangian methods in a nutshell



### The backward pass: differentiating closest feasible QP solutions

$$s^*(\theta) = \arg \min_{s \in \mathbb{R}^n} \frac{1}{2} \|s\|^2 \text{ s.t. } x^*(\theta), z^*(\theta) \in \arg \min_{x \in \mathbb{R}^n} \max_{z \in \mathbb{R}_+^m} L(x, z, s; \theta),$$

$$\text{with } L(x, z, s; \theta) \triangleq f(x; \theta) + z^T (C(\theta)x - u(\theta) - s).$$

**Contribution:**  
QPLayer: A full differentiable pipeline in C++ connected with PyTorch.

A classical technique: **the Implicit Function Theorem.**

$$G(x, z, t; \theta) \triangleq \begin{bmatrix} \nabla_x f(x; \theta) + C(\theta)^T z \\ C(\theta)x - u(\theta) - t \\ [t]_- + z ]_+ - z \\ C(\theta)^T [t]_+ \end{bmatrix}$$

The map is **path-differentiable**.

$$\left( \frac{\partial x^*}{\partial \theta}, \frac{\partial z^*}{\partial \theta} \right) \in \arg \min_w \left\| \frac{\partial G(x^*, z^*; \theta)}{\partial v^*} w + \frac{\partial G(x^*, z^*; \theta)}{\partial \theta} \right\|_2,$$

$$\Pi \frac{\partial t^*}{\partial \theta} \in \frac{\partial s^*}{\partial \theta}, \text{ with } \Pi \in \partial([\cdot]_+)(t^*).$$

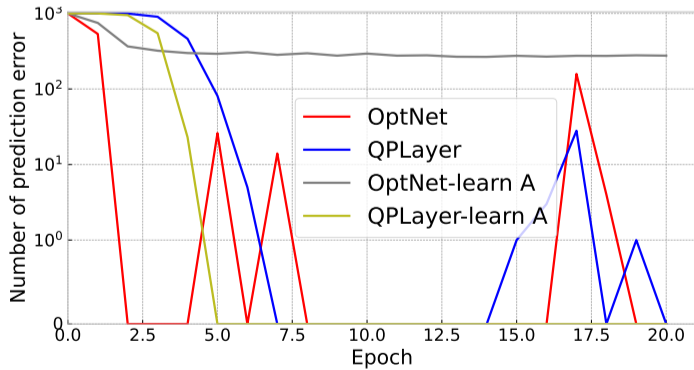
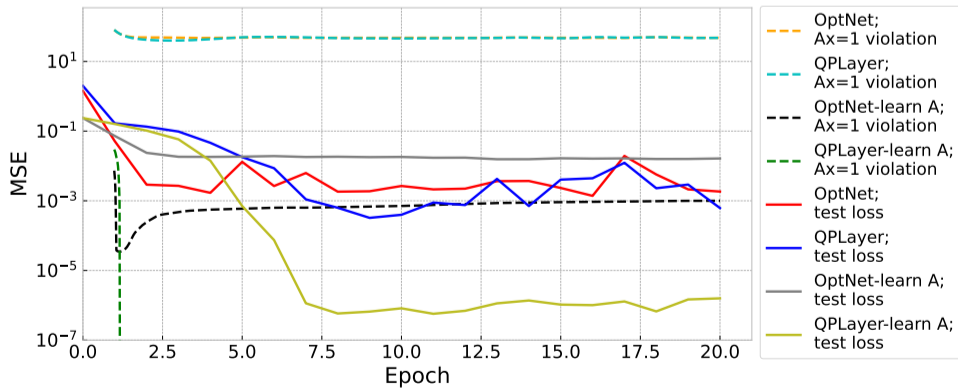
**Contribution:**  
Extend the technique for the closest feasible QP solutions.

**Contribution:**  
Efficient algorithms to solve these problems.

### Benchmarks

|                    | OptNet         | QPLayer      |
|--------------------|----------------|--------------|
| Forward pass (ms)  | 615.84 ± 16.15 | 55.2 ± 6.93  |
| Backward pass (ms) | 61.26 ± 2.84   | 39.27 ± 2.17 |
| Final Loss         | 0.02604        | 0.02556      |

**Table:** Average computational times (over 800 epochs) for solving a cart-pole example with friction when using OptNet or QPLayer. Randomized smoothing is used for obtaining informative gradients [1].



**Figure:** Sudoku training and test plots using QPLayer and OptNet layers. QPLayer can learn LPs, whereas OptNet is restricted to strictly convex QPs, which limits its representational power. QPLayer can also be specialized to learn models satisfying specific linear constraints.

### Future work

- Extension to GPUs,
- Beyond LPs and QPs: SOCPs, SDPs, etc..

### References

1. Lidec, Q. L., Montaut, L., Schmid, C., Laptev, I. & Carpentier, J. *Leveraging Randomized Smoothing for Optimal Control of Nonsmooth Dynamical Systems*. 2022.