Next iterations of quadratic programming for adaptive and robust motion control

Stéphane Caron

December 12, 2023

Inria, École normale supérieure

Motivation

Quadratic programming for stair climbing

LIPM walking controller[1] = revisit Kajita *et al.* with QPs:

- **QP1:** linear model predictive control (ups [Kaj+03])
- **QP2:** wrench distribution (ups [Kaj+10])
- **QP3:** inverse kinematics (ups [Kaj+10])

Two consequences:

- Explicit cost functions
- Behavior switches on constraint saturation

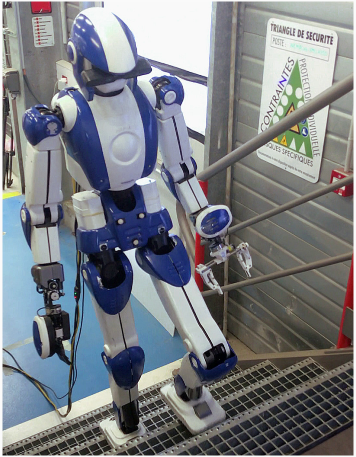
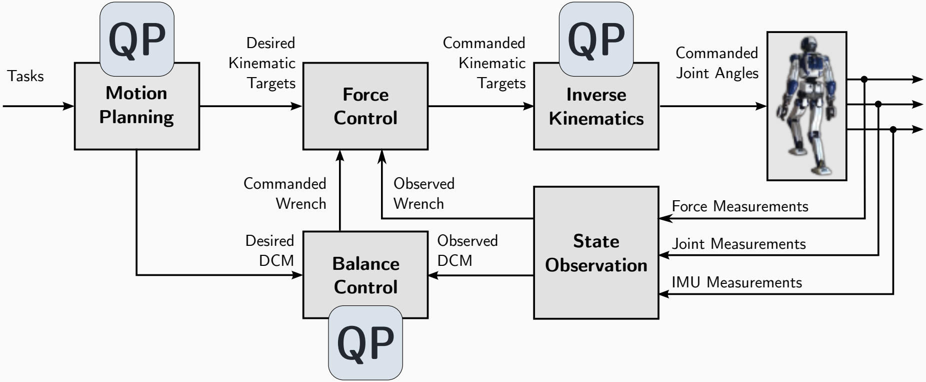


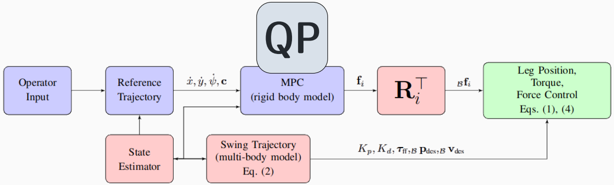
Figure 1: Stair climbing at Airbus

[1] Stéphane Caron, Abderrahmane Kheddar, and Olivier Tempier. "Stair Climbing Stabilization of the HRP-4 Humanoid Robot using Whole-body Admittance Control". In: *IEEE International Conference on Robotics and Automation*. May 2019.

LIPM walking controller pipeline



Quadratic programming for quadrupedal MPC



2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)
Madrid, Spain, October 1-5, 2018

Dynamic Locomotion in the MIT Cheetah 3 Through Convex Model-Predictive Control

Jared Di Carlo[1], Patrick M. Wensing[2], Benjamin Katz[3], Gerardo Bledt[1,3], and Sangbae Kim[3]

***Abstract*—This paper presents an implementation of model predictive control (MPC) to determine ground reaction forces for a torque-controlled quadruped robot. The robot dynamics are simplified to formulate the problem as convex optimization while still capturing the full 3D nature of the system. With the simplified model, ground reaction force planning problems are formulated for prediction horizons of up to 0.5 seconds, and are solved to optimality in under 1 ms at a rate of 20-30 Hz. Despite using a simplified model, the robot is capable of robust locomotion at a variety of speeds. Experimental results demonstrate control of gaits including stand, trot, flying-trot, pronk, bound, pace, a 3-legged gait, and a full 3D gallop. The robot achieved forward speeds of up to 3 m/s, lateral speeds up to 1 m/s, and angular speeds up to 180 deg/sec. Our approach is general enough to perform all these behaviors with the same set of gains and weights.**

I. INTRODUCTION

Control of highly dynamic legged robots is a challenging problem due to the underactuation of the body during many gaits and due to constraints placed on ground reaction forces. As an example, during dynamic gaits[4] such as bounding or galloping, the body of the robot is always underactuated. Additionally, ground reaction forces must always remain in a friction cone to avoid slipping. Current solutions for highly dynamic locomotion include heuristic controllers for hopping and bounding [1], which are effective, but difficult to tune; two-dimensional planar simplifications [2], which are only applicable for gaits without lateral or roll dynamics; and evolutionary optimization for galloping [3], which cannot currently be solved fast enough for online use. Recent results on hardware include execution of bounding limit cycles discovered offline with HyQ [4] and learned pronking, trotting, and bounding gaits on StarlETH [5].

Predictive control can stabilize these dynamic gaits by anticipating periods of flight or underactuation, but is difficult to solve due to the nonlinear dynamics of legged robots and the large number of states and control inputs. Nonlinear optimization has been shown to be effective for predictive control of hopping robots [6], humanoids [7], [8], and quadrupeds [9], with [9] demonstrating the utility of heuristics to regularize the optimization. Another common approach is to use both a high-level planner, such as in [10], [11] and a lower level controller to track the plan. More

Authors are with the [1]Department of Electrical Engineering and Computer Science at the Massachusetts Institute of Technology, Cambridge, MA, 02139, USA; the [2]Department of Aerospace and Mechanical Engineering at the University of Notre Dame, Notre Dame, IN, 46556; and the [3]Department of Mechanical Engineering at the Massachusetts Institute of Technology, Cambridge, MA, 02139, USA. email: dicarloj@mit.edu, pwensing@nd.edu, benkatz@mit.edu, gbledt@mit.edu, sangbae@mit.edu

[4]In this paper, the term *dynamic gaits* is used to refer to gaits with significant periods of flight or underactuation.

Fig. 1. The MIT Cheetah 3 Robot galloping at 2.5 m/s

recently, the experimental results in [12] show that whole-body nonlinear MPC can be used to stabilize trotting and jumping.

The stabilization of the quadruped robot HyQ using convex optimization discussed in [13] demonstrates the utility of convex optimization, but the approach cannot be immediately extended to dynamic gaits due to the quasi-static simplifications made to the robot model. Similarly, in bipedal locomotion, convex optimization has been used to find the best forces to satisfy instantaneous dynamics requirements [14] and to plan footsteps with the linear inverted pendulum model [15] but the latter approach does not include orientation in the predictive model.

While galloping is well studied in the field of biology [16], [17], surprisingly few hardware implementations of galloping exist. The first robot to demonstrate galloping was the underactuated quadruped robot Scout II [18], which reached 1.3 m/s, but had limited control of yaw. The MIT Cheetah 1 robot [19] achieved high-speed galloping, but was constrained to a plane. To the best of our knowledge, the only previous implementation of a fully 3D gallop with yaw control is on the hydraulically actuated WildCat robot [20], developed by Boston Dynamics. Unfortunately, no specific details about WildCat or its control system have been published.

The main contribution of this paper is a predictive controller which stabilizes a large number of gaits, including those with complex orientation dynamics. On hardware, we achieved a maximum yaw rate of 180 deg/sec and a maximum linear velocity of 3.0 m/s during a fully 3D gallop, which we believe to be the fastest gallop of an electrically actuated robot, and the fastest angular velocity of any legged robot similar in scale to Cheetah 3. Our controller can be formulated as a single convex optimization problem which considers a 3D, 12 DoF model of the robot. The solution of

This work was supported by the National Science Foundation [NSF-IIS-1350879] and the Air Force Office of Scientific Research [FA2386-17-1-4661]

978-1-5386-8094-0/18/$31.00 ©2018 IEEE 7440

Linearize model predictive control QP:

- **Cost:** trajectory tracking + input regularization
- **Equality constraints:** no force on swing feet
- **Inequality constraints:** friction cones

[2]Jared Di Carlo, Patrick M Wensing, Benjamin Katz, Gerardo Bledt, and Sangbae Kim. "Dynamic locomotion in the mit cheetah 3 through convex model-predictive control". In: *IEEE/RSJ international conference on intelligent robots and systems.* 2018.

Why bother with QP?

Why bother with quadratic programming?

Cons:

- These works are oldies!
- Whole-body model predictive control has become feasible [Dan+22; KO22; Gra+23]
- RL-trained policies can achieve better robustness [Lee+20; Kum+21]

Pros:

- Quadratic programming is improving: performance, differentiability, ...
- SQP with "one Newton step per cycle" = QP
- Quadratic programming not necessarily $\perp$ to nonlinear OC and RL

Three areas of improvement

1. Runtime and accuracy
2. Handling infeasibility
3. Differentiability

Runtime and accuracy

Quadratic programming

A quadratic program can be generally written as:

$$\begin{aligned} \underset{x}{\text{minimize}} \quad & \frac{1}{2} x^T P x + q^T x \\ \text{subject to} \quad & G x \leq h \\ & A x = b \\ & lb \leq x \leq ub \end{aligned}$$

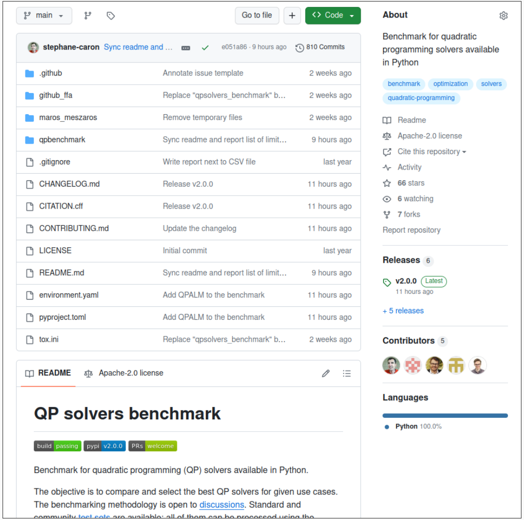
For example in Python:

```python
from qpsolvers import solve_qp

M = np.array([[1., 2., 0.], [-8., 3., 2.], [0., 1., 1.]])
P = M.T @ M  # this is a positive definite matrix
q = np.array([3., 2., 3.]) @ M
G = np.array([[1., 2., 1.], [2., 0., 1.], [-1., 2., -1.]])
h = np.array([3., 2., -2.])

x = solve_qp(P, q, G, h, solver="proxqp")
```

Setup: `pip install qpsolvers[open_source_solvers]`

Open source QP solvers

Solver name	Algorithm	Warm-start	Year
CVXOPT	Interior point	-	2013
ECOS	Interior point	-	2013
qpOASES	Active set	yes	2014
quadprog	Active set	-	2015
SCS	Augmented Lagrangian	yes	2016
HPIPM	Interior point	yes	2017
HiGHS	Active set	-	2017
OSQP	Augmented Lagrangian	yes	2017
DAQP	Active set	-	2021
qpSWIFT	Interior point	-	2021
Clarabel	Interior point	-	2022
QPALM	Augmented Lagrangian	yes	2022
ProxQP	Augmented Lagrangian	yes	2022
PIQP	Proximal interior point	-	2023



main | Go to file | + | Code

stephane-caron Sync readme and ... e051a86 · 9 hours ago 810 Commits

.github	Annotate issue template	2 weeks ago
github_ffa	Replace "qpsolvers_benchmark" b...	2 weeks ago
maros_meszaros	Remove temporary files	2 weeks ago
qpbenchmark	Sync readme and report list of limit...	9 hours ago
.gitignore	Write report next to CSV file	last year
CHANGELOG.md	Release v2.0.0	11 hours ago
CITATION.cff	Release v2.0.0	11 hours ago
CONTRIBUTING.md	Update the changelog	11 hours ago
LICENSE	Initial commit	last year
README.md	Sync readme and report list of limit...	9 hours ago
environment.yaml	Add QPALM to the benchmark	11 hours ago
pyproject.toml	Add QPALM to the benchmark	11 hours ago
tox.ini	Replace "qpsolvers_benchmark" b...	2 weeks ago

README Apache-2.0 license

QP solvers benchmark

build passing | pypi v2.0.0 | PRs welcome

Benchmark for quadratic programming (QP) solvers available in Python.

The objective is to compare and select the best QP solvers for given use cases. The benchmarking methodology is open to discussions. Standard and community test sets are available: all of them can be processed using the

About

Benchmark for quadratic programming solvers available in Python

benchmark optimization solvers quadratic-programming

Readme
Apache-2.0 license
Cite this repository
Activity
66 stars
6 watching
7 forks
Report repository

Releases 6

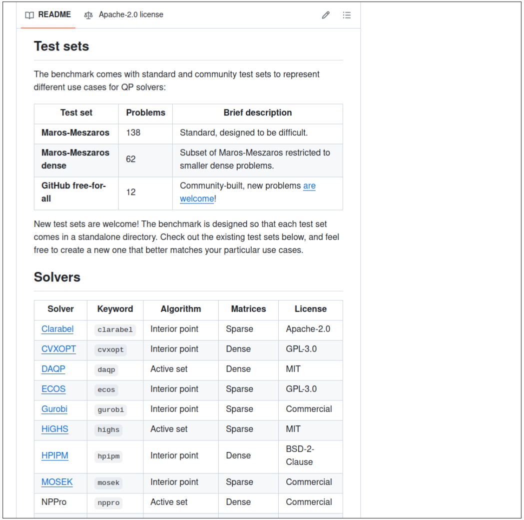
v2.0.0 Latest
11 hours ago

+ 5 releases

Contributors 5

Languages

Python 100.0%



README Apache-2.0 license

Test sets

The benchmark comes with standard and community test sets to represent different use cases for QP solvers:

Test set	Problems	Brief description
Maros-Meszaros	138	Standard, designed to be difficult.
Maros-Meszaros dense	62	Subset of Maros-Meszaros restricted to smaller dense problems.
GitHub free-for-all	12	Community-built, new problems are welcome!

New test sets are welcome! The benchmark is designed so that each test set comes in a standalone directory. Check out the existing test sets below, and feel free to create a new one that better matches your particular use cases.

Solvers

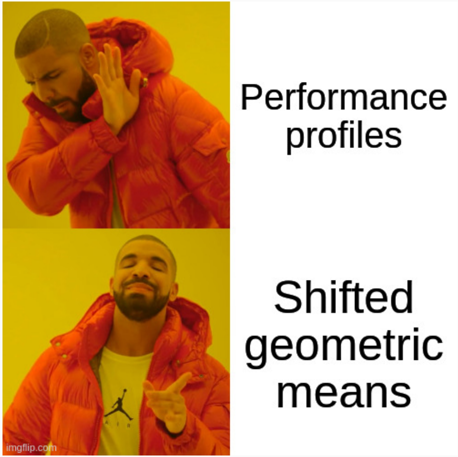
Solver	Keyword	Algorithm	Matrices	License
Clarabel	`clarabel`	Interior point	Sparse	Apache-2.0
CVXOPT	`cvxopt`	Interior point	Dense	GPL-3.0
DAQP	`daqp`	Active set	Dense	MIT
ECOS	`ecos`	Interior point	Sparse	GPL-3.0
Gurobi	`gurobi`	Interior point	Sparse	Commercial
HiGHS	`highs`	Active set	Sparse	MIT
HPIPM	`hpipm`	Interior point	Dense	BSD-2-Clause
MOSEK	`mosek`	Interior point	Sparse	Commercial
NPPro	`nppro`	Active set	Dense	Commercial

GitHub: *`https://github.com/qpsolvers/qpbenchmark`*

How to compare solvers?

How can we compare solver performances over whole test sets?

Summary of existing expert knowledge



[3] Hans Mittelmann. *Benchmarks for optimization software*. Sept. 8, 2019. URL: `http://plato.asu.edu/bench.html`.

Shifted geometric means

The shifted geometric mean of a series $T^s = (T_i)_{i=1}^n$ is:

$$\mathrm{shgeom}(T^s) = \sqrt[n]{\prod_i (T_i^s + k)} - k$$

Scaled shifted geometric mean: $\overline{T}^s = T^s / \operatorname{arg\,min}_s T^s$.

Interpretation

A solver with a scaled $\mathrm{shgeom}(\mathit{runtimes}) = Y$ is $Y\times$ slower than the best solver.

[4] Hans Mittelmann. *Benchmarks for optimization software*. Sept. 8, 2019. URL: `http://plato.asu.edu/bench.html`.

Don’t trust performance profiles

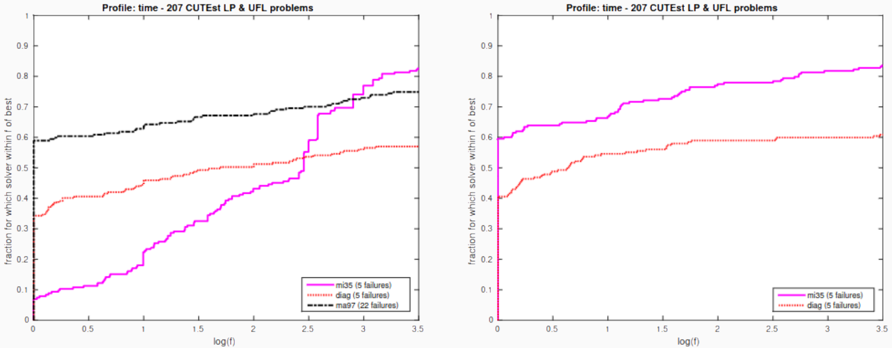


Figure 2: Time performance profiles for a real test case. Left: three solvers. Right: two solvers.

[5] Nicholas Gould and Jennifer Scott. “A note on performance profiles for benchmarking software”. In: *ACM Transactions on Mathematical Software* 43.2 (2016).

Optimality conditions

Let x^*, y^*, z^* denote a primal-dual solution to our QP:

- **Primal residual:** $r_p := \max(\|Ax^* - b\|_\infty, [Gx^* - h]_+)$
- **Dual residual:** $r_d := \|Px^* + q + A^T y^* + G^T z^*\|_\infty$
- **Duality gap:** $r_g := |{x^*}^T P x^* + q^T x^* + b^T y^* + h^T z^*|$

Optimality

The solution (x^*, y^*, z^*) returned by a QP solver is optimal *if and only if* $r_p = 0, r_d = 0, r_g = 0$.

Accuracy of a solution

Accuracy of a solution with absolute ϵ_{abs} and relative ϵ_{rel} tolerances:

$$
\begin{aligned}
r_p &\leq \epsilon_{abs} + \epsilon_{rel} \max(\|Ax^*\|_\infty, \|Gx^*\|_\infty, \|b\|_\infty, \|h\|_\infty) \\
r_d &\leq \epsilon_{abs} + \epsilon_{rel} \max(\|Px^*\|_\infty, \|q\|_\infty, \|A^T y^*\|_\infty, \|G^T z^*\|_\infty) \\
r_g &\leq \epsilon_{abs} + \epsilon_{rel} \max(|x^{*T} P x^*|, |q^T x^*|, |b^T y^*|, |h^T z^*|)
\end{aligned}
$$

Contract from the QP solver.[6]

Default solver accuracies vary a lot: solvers **trade off** runtime and accuracy.

[6] OSQP does not check the duality gap and may return false solutions, such as [ODo21, Section 7.2].

Test sets

Check results for the test sets that interest you:

- **Model predictive control:** humanoid, quadruped, wheeled biped, ...
  https://github.com/qpsolvers/mpc_qpbenchmark
- **Maros-Meszaros:** standard, 138 difficult problems.
  https://github.com/qpsolvers/maros_meszaros_qpbenchmark
- **Free-for-all:** open to all problems, no restriction.
  https://github.com/qpsolvers/free_for_all_qpbenchmark

Propose your own

The benchmark is a `qpbenchmark` command-line tool: try your own use cases!

Setup: `pip install qpbenchmark`

Open questions

Goal of the benchmark:

- What are the best solvers for task $\in$ { MPC, IK, ID, ... }?
- What is the ideal runtime/accuracy tradeoff for each task?
- What runtime/accuracy tradeoff can each solver achieve?

Current limitations:

- Cold-start only (#101)
- CPU thermal throttling (#88)

Handling infeasibility

Recursive feasibility

Guarantee that the optimization problem at the next cycle is feasible.

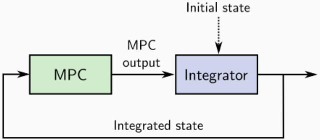


Figure 3: **Open-loop MPC:** strong RF doable [CWF17].

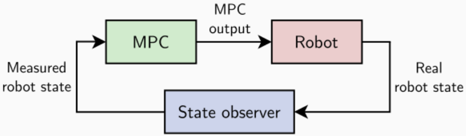


Figure 4: **Closed-loop MPC:** no recursive feasibility?

ProxQP

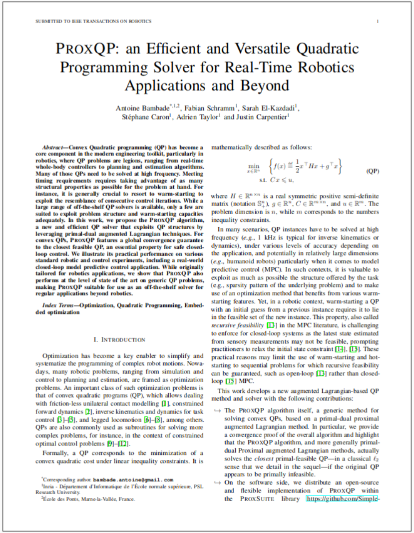
- Handles semidefinite P (including $P = 0$)
- Warm- and hot-starting for *e.g.* MPC
- Competitive runtime and accuracy perfs
- (Theoretical: global convergence guarantee.)

Property

ProxQP converges to the solution of:

- The QP itself, if it is feasible, or
- The *closest feasible QP* otherwise.

Never fails: great for real-time control.


SUBMITTED TO IEEE TRANSACTIONS ON ROBOTICS 1

PROXQP: an Efficient and Versatile Quadratic Programming Solver for Real-Time Robotics Applications and Beyond

Antoine Bambade[*,1,2], Fabian Schramm[1], Sarah El-Kazdadi[1], Stéphane Caron[1], Adrien Taylor[1] and Justin Carpentier[1]

***Abstract*—Convex Quadratic programming (QP) has become a core component in the modern engineering toolkit, particularly in robotics, where QP problems are legions, ranging from real-time whole-body controllers to planning and estimation algorithms. Many of those QPs need to be solved at high frequency. Meeting timing requirements requires taking advantage of as many structural properties as possible for the problem at hand. For instance, it is generally crucial to resort to warm-starting to exploit the resemblance of consecutive control iterations. While a large range of off-the-shelf QP solvers is available, only a few are suited to exploit problem structure and warm-starting capacities adequately. In this work, we propose the PROXQP algorithm, a new and efficient QP solver that exploits QP structures by leveraging primal-dual augmented Lagrangian techniques. For convex QPs, PROXQP features a global convergence guarantee to the closest feasible QP, an essential property for safe closed-loop control. We illustrate its practical performance on various standard robotic and control experiments, including a real-world closed-loop model predictive control application. While originally tailored for robotics applications, we show that PROXQP also performs at the level of state of the art on generic QP problems, making PROXQP suitable for use as an off-the-shelf solver for regular applications beyond robotics.**

***Index Terms*—Optimization, Quadratic Programming, Embedded optimization**

I. INTRODUCTION

Optimization has become a key enabler to simplify and systematize the programming of complex robot motions. Nowadays, many robotic problems, ranging from simulation and control to planning and estimation, are framed as optimization problems. An important class of such optimization problems is that of convex quadratic programs (QP), which allows dealing with friction-less unilateral contact modelling [1], constrained forward dynamics [2], inverse kinematics and dynamics for task control [3]–[5], and legged locomotion [6]–[8], among others. QPs are also commonly used as subroutines for solving more complex problems, for instance, in the context of constrained optimal control problems [9]–[12].

Formally, a QP corresponds to the minimization of a convex quadratic cost under linear inequality constraints. It is mathematically described as follows:

$$\min_{x \in \mathbb{R}^n} \left\{ f(x) \stackrel{\text{def}}{=} \frac{1}{2} x^\top H x + g^\top x \right\} \quad \text{s.t. } Cx \leqslant u, \tag{QP}$$

where $H \in \mathbb{R}^{n \times n}$ is a real symmetric positive semi-definite matrix (notation $\mathbb{S}^n_+$), $g \in \mathbb{R}^n$, $C \in \mathbb{R}^{m \times n}$, and $u \in \mathbb{R}^m$. The problem dimension is n, while m corresponds to the numbers inequality constraints.

In many scenarios, QP instances have to be solved at high frequency (*e.g.*, 1 kHz is typical for inverse kinematics or dynamics), under various levels of accuracy depending on the application, and potentially in relatively large dimensions (*e.g.*, humanoid robots) particularly when it comes to model predictive control (MPC). In such contexts, it is valuable to exploit as much as possible the structure offered by the task (e.g., sparsity pattern of the underlying problem) and to make use of an optimization method that benefits from various warm-starting features. Yet, in a robotic context, warm-starting a QP with an initial guess from a previous instance requires it to lie in the feasible set of the new instance. This property, also called *recursive feasibility* [13] in the MPC literature, is challenging to enforce for closed-loop systems as the latest state estimated from sensory measurements may not be feasible, prompting practitioners to relax the initial state constraint [14], [13]. These practical reasons may limit the use of warm-starting and hot-starting to sequential problems for which recursive feasibility can be guaranteed, such as open-loop [13] rather than closed-loop [13] MPC.

This work develops a new augmented Lagrangian-based QP method and solver with the following contributions:

↪ The PROXQP algorithm itself, a generic method for solving convex QPs, based on a primal-dual proximal augmented Lagrangian method. In particular, we provide a convergence proof of the overall algorithm and highlight that the PROXQP algorithm, and more generally primal-dual Proximal augmented Lagrangian methods, actually solves the *closest* primal-feasible QP—in a classical ℓ_2 sense that we detail in the sequel—if the original QP appears to be primally infeasible.

↪ On the software side, we distribute an open-source and flexible implementation of PROXQP within the PROXSUITE library https://github.com/Simple-

[*]Corresponding author. bambade.antoine@gmail.com
[1]Inria - Département d'Informatique de l'École normale supérieure, PSL Research University.
[2]École des Ponts, Marne-la-Vallée, France.

[7] Antoine Bambade, Fabian Schramm, Sarah El Kazdadi, Stéphane Caron, Adrien Taylor, and Justin Carpentier. "PROXQP: an Efficient and Versatile Quadratic Programming Solver for Real-Time Robotics Applications and Beyond". working paper or preprint. Sept. 2023. URL: *https://inria.hal.science/hal-04198663*.

ProxQP on MPC problems

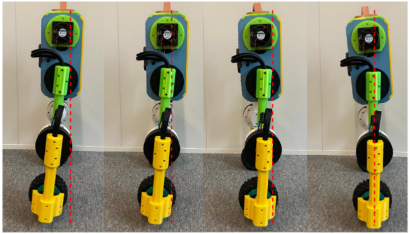
Linearize model predictive control as a QP:

$$\min_{\substack{x_t \in \mathbb{R}^{n_x} \\ u_t \in \mathbb{R}^{n_u}}} w_T \|x_T - x_{\text{goal}}\|_2^2 + \sum_{t=0}^{T-1} w_x \|x_t - x_{\text{goal}}\|_2^2 + w_u \|u_t\|_2^2,$$
$$\text{s.t. } x_{t+1} = A x_t + B u_t, \; x_0 = x_{\text{init}},$$
$$C x_t + D u_t \leqslant e_t,$$

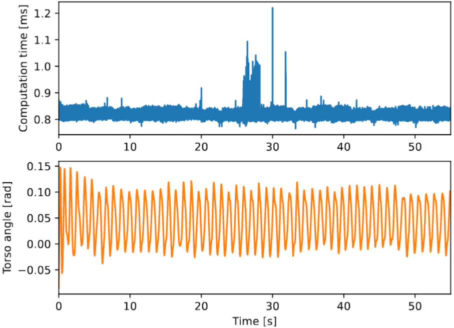
- Dimensions: $n = 50$, $m = 100$
- Time: $T = 50$ steps, $\mathrm{d}t = 20$ ms
- CPU: Raspberry Pi 4 (1.8 GHz) single-core
- Hot-starting: 0.8 ± 0.02 ms

TABLE V: Humanoid locomotion MPC problems with perturbations.

Noise Level	ProxQP	quadprog	OSQP	qpOASES	SCS	qpSWIFT	MOSEK
10.0	**11.9±9.7%**	1.0±0.2%	1.9±0.9%	1.0±0.2%	1.0±0.2%	0±0.0%	1.0±0.2%
5.0	**58.38±36.4%**	1.1±0.3%	2.1±1.1%	1.1±0.3%	1.1±0.4%	0±0.0%	1.1±0.3%
1.0	**100±0.0%**	1.4±0.8%	3.5±2.4%	1.4±0.8%	1.5±1.1%	0±0.0%	1.4±0.9%
0.5	**100±0.0%**	1.8±1.2%	5.5±3.8%	1.9±1.5%	2.1±1.6%	0±0.0%	1.8±1.2%
0.1	**100±0.0%**	3.3±2.6%	51.6±36.7%	4.3±3.8%	4.9±4.3%	0±0.0%	3.3±2.6%
0.05	**100±0.0%**	3.5±3.2%	97.6±13.5%	5.0±6.9%	7.7±6.5%	0±0.0%	3.5±3.2
0.01	**100±0.0%**	4.4±4.4%	**100±0.0%**	7.7±9.8%	60.2±37.8%	0±0.0%	4.4±4.5%
10^{-3}	**100±0.0%**	5.0±5.2%	**100±0.0%**	11.4±12.5%	**100±0.0%**	0±0.0%	5.0±5.2%
10^{-4}	**100±0.0%**	5.0±5.2%	**100±0.0%**	15.5±16.8%	**100±0.0%**	0±0.0%	5.0±5.2%
10^{-5}	**100±0.0%**	5.0±5.2%	**100±0.0%**	83.0±36.5%	99.1±8.9%	0±0.0%	5.1±5.3%
10^{-7}	**100±0.0%**	5.0±5.2%	**100±0.0%**	**100±0.0%**	97±14.8%	0±0.0%	44.8±34.2%
10^{-9}	**100±0.0%**	5.0±5.2%	**100±0.0%**	**100±0.0%**	**100±0.0%**	0±0.0%	**100±0.0%**
0.0	**100±0.0%**	**100±0.0%**	**100±0.0%**	**100±0.0%**	**100±0.0%**	0±0.0%	**100±0.0%**



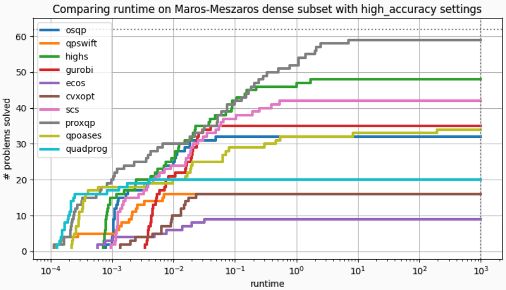
QP solve times on the embedded system



ProxSuite

THE ADVANCED PROXIMAL OPTIMIZATION TOOLBOX

- **Fast:** C++ with custom linear Cholesky solver
- **Backends:** dense, sparse, matrix-free optim.
- **Easy-to-use:** standard API, Python/Julia
- **Open-source:** BSD-license, Conda/PyPI



Setup: `conda install -c conda-forge proxsuite` / `pip install proxsuite`

∇ THROUGH INFEASIBLE PROBLEMS

Use convex QP as a deep learning layer: [8]

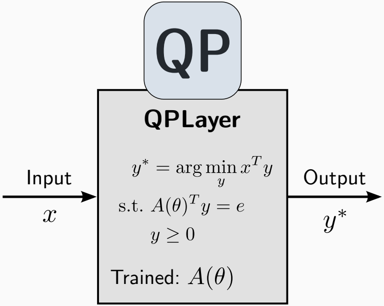


Figure 5: Learning to play Sudoku: the layer is trainable *if and only if* we can ∇ through infeasible LPs.

[8] Brandon Amos and J Zico Kolter. "Optnet: Differentiable optimization as a layer in neural networks". In: *International Conference on Machine Learning*. PMLR. 2017, pp. 136–145.

Example: making an LP feasible

Solve the *closest feasible* QP, and penalize in the learning loss its current distance w.r.t. the space of feasible QPs.

input

QP layer

$$\min_{x \in \mathbb{R}^n} -\text{input}^\mathrm{T} x \quad \text{s.t.} \begin{cases} Ax = 1, \\ x \geq 0. \end{cases}$$

Forward pass

x^*, s^*

$\dfrac{\partial L}{\partial x^*} \; \dfrac{\partial L}{\partial s^*}$

Backward pass

$$L(x^*, s^*) = \frac{1}{2}\|x^* - x^{\text{true}}\|_2^2 + 10^{-3}\|s^*\|_2^2$$

Remark this distance can be measured with some vector s^*, defined later.

Robotics example

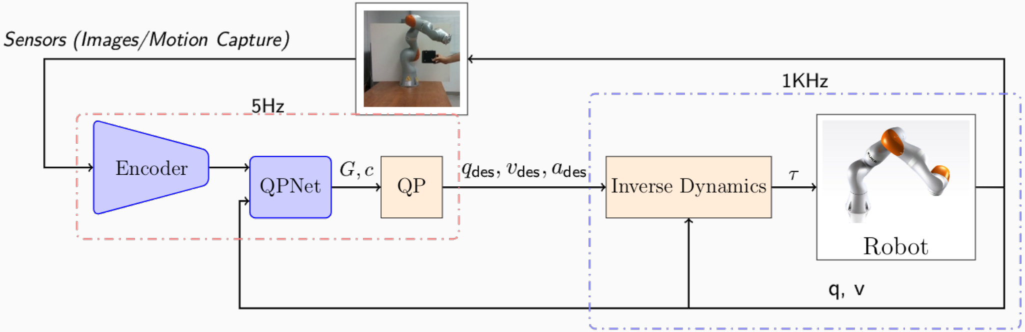


Figure 6: Differentiable pipeline training a QP-based motion policy with visual inputs.[9]

[9] Avadesh Meduri, Huaijiang Zhu, Armand Jordana, and Ludovic Righetti. “MPC with Sensor-Based Online Cost Adaptation”. In: *2023 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE. 2023, pp. 996–1002.

QPLayer methodology

Forward pass: solving the *closest feasible* QP - formulated hierarchically

$$s^\star(\theta) = \arg\min_{s\in\mathbb{R}^{n_i}} \frac{1}{2}\|s\|_2^2 \quad \text{s.t. } x^\star(\theta), z^\star(\theta) \in \arg\min_{x\in\mathbb{R}^n} \max_{z\in\mathbb{R}_+^{n_i}} L(x,z,s;\theta), \tag{QP-H(θ)}$$

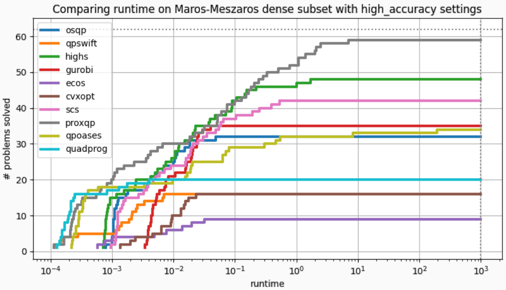
- **automatic:** make use of a property of augmented Lagrangians via ProxQP
- we get the infeasibility gap $s^\star(\theta)$ **(= 0 if feasible)**

[10] Antoine Bambade, Fabian Schramm, Adrien Taylor, and Justin Carpentier. "QPLayer: efficient differentiation of convex quadratic optimization". working paper or preprint. June 2023. URL: *https://inria.hal.science/hal-04133055*.

ProxSuite

THE ADVANCED PROXIMAL OPTIMIZATION TOOLBOX

- Fast: C++ with custom linear Cholesky solver
- Backends: dense, sparse, matrix-free optim.
- Easy-to-use: standard API, Python/Julia
- Open-source: BSD-license, Conda/PyPI
- **QPLayer: included**, freshly baked!



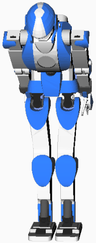
Setup: `conda install -c conda-forge proxsuite` / `pip install proxsuite`

Conclusion

Conclusion

- **QP in control pipelines:** model predictive control, inverse dynamics, ...
- **QP benchmark:** evaluate runtimes and accuracy, open to new problems
- **ProxQP:** handle infeasibility, real-time control
- **QPLayer:** ∇ through QP layers, *everywhere*

That's all folks!



Bibliography

[AK17] Brandon Amos and J Zico Kolter. “Optnet: Differentiable optimization as a layer in neural networks”. In: *International Conference on Machine Learning*. PMLR. 2017, pp. 136–145.

[Bam+23a] Antoine Bambade, Fabian Schramm, Sarah El Kazdadi, Stéphane Caron, Adrien Taylor, and Justin Carpentier. “PROXQP: an Efficient and Versatile Quadratic Programming Solver for Real-Time Robotics Applications and Beyond”. working paper or preprint. Sept. 2023. URL: `https://inria.hal.science/hal-04198663`.

[Bam+23b] Antoine Bambade, Fabian Schramm, Adrien Taylor, and Justin Carpentier. “QPLayer: efficient differentiation of convex quadratic optimization”. working paper or preprint. June 2023. URL: `https://inria.hal.science/hal-04133055`.

[CKT19] Stéphane Caron, Abderrahmane Kheddar, and Olivier Tempier. “Stair Climbing Stabilization of the HRP-4 Humanoid Robot using Whole-body Admittance Control”. In: *IEEE International Conference on Robotics and Automation*. May 2019.

[CWF17] Matteo Ciocca, Pierre-Brice Wieber, and Thierry Fraichard. “Strong recursive feasibility in model predictive control of biped walking”. In: *2017 IEEE-RAS 17th International Conference on Humanoid Robotics (Humanoids)*. IEEE. 2017, pp. 730–735.

[Dan+22] Ewen Dantec, Maximilien Naveau, Pierre Fernbach, Nahuel Villa, Guilhem Saurel, Olivier Stasse, Michel Taix, and Nicolas Mansard. “Whole-Body Model Predictive Control for Biped Locomotion on a Torque-Controlled Humanoid Robot”. In: *2022 IEEE-RAS 21st International Conference on Humanoid Robots (Humanoids)*. 2022.

[Di +18] Jared Di Carlo, Patrick M Wensing, Benjamin Katz, Gerardo Bledt, and Sangbae Kim. “Dynamic locomotion in the mit cheetah 3 through convex model-predictive control”. In: *IEEE/RSJ international conference on intelligent robots and systems*. 2018.

[Gra+23] Ruben Grandia, Fabian Jenelten, Shaohui Yang, Farbod Farshidian, and Marco Hutter. "Perceptive Locomotion Through Nonlinear Model-Predictive Control". In: *IEEE Transactions on Robotics* (2023).

[GS16] Nicholas Gould and Jennifer Scott. "A note on performance profiles for benchmarking software". In: *ACM Transactions on Mathematical Software* 43.2 (2016).

[Kaj+03] Shuuji Kajita, Fumio Kanehiro, Kenji Kaneko, Kiyoshi Fujiwara, Kensuke Harada, Kazuhito Yokoi, and Hirohisa Hirukawa. "Biped walking pattern generation by using preview control of zero-moment point". In: *IEEE International Conference on Robotics and Automation*. 2003.

[Kaj+10] Shuuji Kajita, Mitsuharu Morisawa, Kanako Miura, Shin'ichiro Nakaoka, Kensuke Harada, Kenji Kaneko, Fumio Kanehiro, and Kazuhito Yokoi. "Biped walking stabilization based on linear inverted pendulum tracking". In: *IEEE/RSJ International Conference on Intelligent Robots and Systems*. 2010.

[KO22] Sotaro Katayama and Toshiyuki Ohtsuka. "Lifted contact dynamics for efficient optimal control of rigid body systems with contacts". In: *2022 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE. 2022, pp. 8879–8886.

[Kum+21] Ashish Kumar, Zipeng Fu, Deepak Pathak, and Jitendra Malik. "Rma: Rapid motor adaptation for legged robots". In: *arXiv preprint arXiv:2107.04034* (2021).

[Lee+20] Joonho Lee, Jemin Hwangbo, Lorenz Wellhausen, Vladlen Koltun, and Marco Hutter. "Learning quadrupedal locomotion over challenging terrain". In: *Science robotics* 5.47 (2020).

[Med+23] Avadesh Meduri, Huaijiang Zhu, Armand Jordana, and Ludovic Righetti. "MPC with Sensor-Based Online Cost Adaptation". In: *2023 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE. 2023, pp. 996–1002.

[Mit19] Hans Mittelmann. *Benchmarks for optimization software*. Sept. 8, 2019. URL: http://plato.asu.edu/bench.html.

[ODo21] Brendan O'Donoghue. "Operator splitting for a homogeneous embedding of the linear complementarity problem". In: *SIAM Journal on Optimization* 31.3 (2021), pp. 1999–2023.