ProxNLP: a primal-dual augmented Lagrangian solver for nonlinear programming in Robotics and beyond

Wilson Jallet[a,b,*], Antoine Bambade[b,c], Nicolas Mansard[a] and Justin Carpentier[b]

***Abstract*— Mathematical optimization is the workhorse behind several aspects of modern robotics and control. In these applications, the focus is on constrained optimization, and the ability to work on manifolds (such as the classical matrix Lie groups), along with a specific requirement for robustness and speed. In recent years, augmented Lagrangian methods have seen a resurgence due to their robustness and flexibility, their connections to (inexact) proximal-point methods, and their interoperability with Newton or semismooth Newton methods. In the sequel, we present primal-dual augmented Lagrangian method for inequality-constrained problems on manifolds, which we introduced in our recent work, as well as an efficient C++ implementation suitable for use in robotics applications and beyond.**

***Paper Type* – Recent Work [1] under review. (Extended with open-sourced implementation)**

I. Introduction

The setting of optimization on manifolds is of great interest in the field of robotics, where *generalized coordinates* are naturally represented using Lie groups [2]. Further, solvers for robotics need to account for physical constraints such as joint angle and torque limits as well as friction cones, but also for task-based constraints which could replace penalties or costs. Problems such as trajectory optimization or inverse dynamics with various task and physical constraints are naturally expressed as nonlinear programs (NLP).

A generic nonlinear program on a manifold $\mathcal{M}$ reads as follows:

$$\begin{aligned} &\min_{x\in\mathcal{M}} f(x) \\ &\text{s.t. } c(x) \in \mathcal{C} \end{aligned} \tag{1}$$

where $c\colon \mathcal{M} \to \mathbb{R}^m$ is a (potentially nonlinear) mapping and $\mathcal{C} \subset \mathbb{R}^m$ is the constraint set.

Equality and inequality-constrained case: We consider the following generic problem, which captures most problems in nonlinear optimization, including in robotics:

$$\min_{x\in\mathcal{M}} f(x) \text{ s.t. } \; g(x) = 0,\; h(x) \leqslant 0. \tag{2}$$

Most problems of interest in robotics can be expressed this way: dynamics as equality constraints, target reaching, obstacle avoidance and friction cones as inequality constraints.

Our proposed approach is based on the augmented Lagrangian method of multipliers [3]–[5], and its primal-dual variant [6]. It was first introduced in[1] [1] where we provide an application to constrained numerical optimal control with a novel variant of the differential dynamic programming (DDP) algorithm. The applicability of augmented Lagrangians to equality-constrained DDP was recently investigated in the robotics literature [7], [8], with an extension to multiple-shooting implicit dynamics in [9].

Overall, our key contribution is an open-source C++ solver for constrained optimization on manifolds for robotics, named `proxnlp`[2], which relies on a novel variant of the augmented Lagrangian method.

II. Methodology

A. Generalized primal-dual augmented Lagrangians

This approach was first introduced for equality-constrained problems in [6]. We recently provided an extension to inequality-constrained problems in [1] with an application to constrained DDP. This method was further applied to convex QPs in Bambade et al. [10].

The classical (Hestenes-Powell-Rockafellar) augmented Lagrangian function for the problem (2) reads:

$$\mathcal{L}_\mu(x; y_e, z_e) = f(x) + \tfrac{1}{2\mu}\|g(x)+\mu y_e\|_2^2 + \tfrac{1}{2\mu}\|[h(x)+\mu z_e]_+\|. \tag{3}$$

Augmented Lagrangians are known to be *exact* penalty functions for constrained optimization, as in their exists an estimate $(\bar{y}, \bar{z})$ and penalty parameter $\bar{\mu} > 0$ such that a minimizer x^* of $\mathcal{L}_{\bar{\mu}}(\cdot; \bar{y}, \bar{z})$ is a solution of (2).

Method of multipliers: The *method of multipliers* algorithm, consists in iteratively minimizing the augmented Lagrangian and taking a (projected) dual ascent step in the multipliers:

$$\begin{aligned} x^{l+1} &= \arg\min_x \mathcal{L}_\mu(x; y^l, z^l), \\ y^{l+1} &= y_e + \tfrac{1}{\mu} g(x^{l+1}) \\ z^{l+1} &= [z_e + \tfrac{1}{\mu} h(x^{l+1})]_+ \end{aligned} \tag{4}$$

This process can also be seen as a proximal-point algorithm for the dual problem to the initial NLP [11].

Primal-dual function: The *primal-dual* augmented Lagrangian (pdAL) adds an additional penalty term for dual variables:

$$\begin{aligned} &\mathcal{M}_\mu(x, y, z; y_e, z_e) \overset{\text{def}}{=} \mathcal{L}_\mu(x; y_e, z_e) \\ &+ \tfrac{1}{2\mu}\|g(x) + \mu(y_e - y)\|_2^2 + \tfrac{1}{2\mu}\|[h(x) + \mu z_e]_+ - \mu z\|_2^2 \end{aligned} \tag{5}$$

[a] LAAS-CNRS, 7 Avenue du Colonel Roche, F-31400 Toulouse, France
[b] Inria, Département d'informatique de l'ENS, École normale supérieure, CNRS, PSL Research University, Paris, France
[c] ENPC, France, [*]corresponding author: wjallet@laas.fr

[1] Paper under review.

[2] `https://github.com/Simple-Robotics/proxnlp`.

Notice that a stationary point (x^*, y^*, z^*) of $\mathcal{M}_\mu(\cdot; y_e, z_e)$ ends up satisfying the KKT conditions of the iteration (4) where $y^{l+1} = y^*$.

B. The primal-dual Newton step

At a nominal point (x^k, y^k, z^k), the primal-dual (quasi-)Newton step for (5) is given by a system of equations equivalent to

$$\begin{bmatrix} H & A^\top & PB^\top \\ A & -\mu I & 0 \\ PB & 0 & -\mu P \end{bmatrix} \begin{bmatrix} \delta x \\ \delta y \\ \delta z \end{bmatrix} = - \begin{bmatrix} \nabla \mathcal{L}(x^k, y^k) \\ c(x^k) + \mu(y^k - y_e) \\ [h(x^k) + \mu z^k]_+ - \mu z_e \end{bmatrix} \tag{6}$$

where H is an estimation of the Lagrangian Hessian $\nabla^2 \mathcal{L}$, and P is a selection matrix which selects rows of the matrices corresponding to the *active set of constraints* $\mathcal{A}(x^k)$, defined as follows:

$$i \in \mathcal{A}(x) \Leftrightarrow (h(x) + \mu z_e)_i \geqslant 0. \tag{7}$$

As shown in [6], this primal-dual step $(\delta x, \delta y, \delta z)$ is a descent direction for the pdAL function (5).

III. Experiments

For solving generic NLPs, we recently implemented our method in a C++ software library named `proxnlp`. We use Eigen as our linear algebra backend [12]. We provide an interface for rigid-body dynamics and classical matrix Lie groups (e.g. SE(3)) using the Pinocchio [13] library which also provides derivatives [14]. We also provide Python bindings. Another C++ software package specifically dedicated to solving control problems using our variant of the DDP algorithm (exploiting the problem structure for increased efficiency) detailed in [1] is currently under development.

a) Simple barycenter on manifold: `proxnlp` is able to operate on manifolds. We can quickly compute the unconstrained barycenter of a few points: in this case our method reduces to simple Newton or Gauss-Newton iterations. See fig. 1 below.

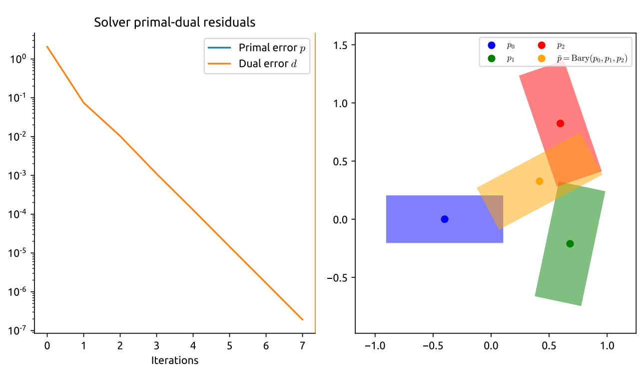

Fig. 1. Barycenter of three 2D poses in the SE(2) Lie group.

b) Double-pendulum: We implemented a simple double-pendulum problem as an NLP using `proxnlp`, Pinocchio and CasADi [15], with a time step of $\Delta t = 30\text{ms}$ and desired convergence threshold of $\epsilon = 10^{-4}$. Here, the second-order derivatives of the dynamics are ignored in the Hessian computation. See fig. 2.

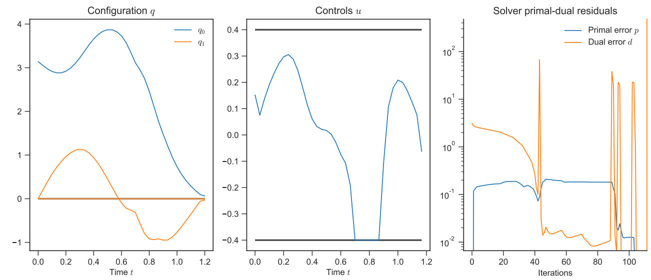

Fig. 2. Angle and torque trajectory of the double-pendulum system, as well as the primal-dual convergence criteria. The controls saturate the imposed limit for a duration of 200 ms.

c) Obstacle avoidance on UR10: This example from our recent preprint [1] was implemented using our experimental code applying the method to DDP. See fig. 3.

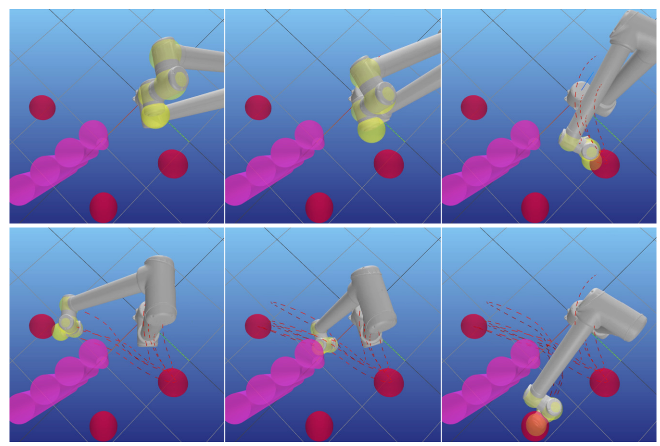

Fig. 3. UR10 reach task. The yellow spheres around the end-effector and wrist links do not collide with the purple cylinders, and the waypoints are reached at the specified times.

d) Pose generation on Talos: See figure 4. The cost function reads, for configuration $q \in \mathcal{Q}$

$$\begin{aligned} J(q) &= 0.1\|q \ominus q_0\|^2 \\ &+ 2.5\|R_{\text{base}}(q) \ominus R_{\text{base}}(q_0)\|^2 \\ &+ 10\|p_{\text{lg}}(q) - p_{\text{rg}}(q) - \mathbf{d}\|^2 \qquad \text{(gripper dist.)} \\ &+ 2\|(p_{\text{le,y}}(q), p_{\text{re,y}}(q)) - (2, -2)\|^2 \qquad \text{(elbow } y\text{)} \\ &+ \|R_{\text{lg}}(q) \ominus R_0\|^2 + \|R_{\text{rg}}(q) \ominus R_1\|^2 \quad \text{(hand orn.)} \end{aligned} \tag{8}$$

where $\mathbf{d} = (0, 0.03, 0)^\top$, lg, rg mean left and right gripper, le, re mean left and right elbow, R_{base} is the body base orientation. We have additional constraints: the right foot must be flat on the ground, the left foot must be $\geqslant 40\,\text{cm}$ above ground with $p_{\text{lf},xy} \in [-0.05, 0.1]$ and a specific orientation, and the right gripper satisfies $p_{\text{rg},y} \leqslant 0, p_{\text{rg},z} \in [-1.1, 1.2]$. The costs and constraints are implemented using CasADi [15].

e) Solo inverse geometry with heightmap: See figure 5. The objective is to generate a feasible pose for the Solo-12

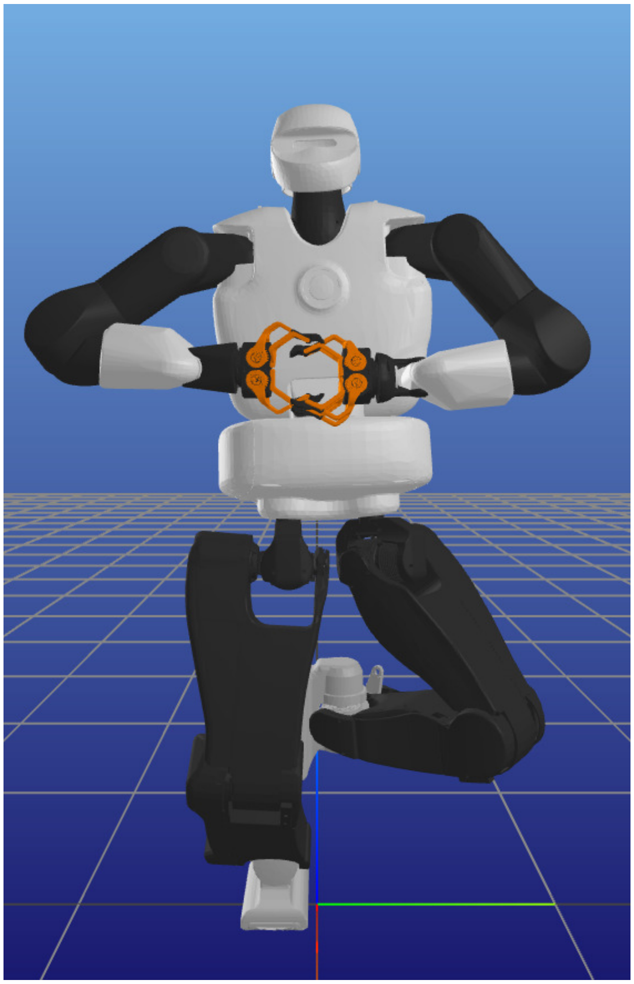

Fig. 4. Generated pose on the Talos robot.

quadruped along with the 3D contact forces at the 4 feet:

$$\min_{q,\{f\}} \|\theta - \theta_0\|^2 + \tfrac{1}{10}\sum_{i=1}^{4} \|f_i\|^2, \tag{9}$$

where θ, θ_0 are the joint angles of the robot (the pose without the base placement). The problem has the following constraints:

- zero angular momentum at the CoM
- the CoM altitude must be higher than the average foot altitude
- contact forces sum to the robot's weight
- contact forces satisfy the friction cone.

Acknowledgements

We would like to thank Fabian Schramm and Guilhem Saurel for their help on testing and packaging the software, and Alessandro Assirelli for providing the pose generation examples on Talos and Solo.

This work was supported in part by the HPC resources from GENCI-IDRIS (Grant AD011011342), the French government under management of Agence Nationale de la Recherche as part of the "Investissements d'avenir" program, reference ANR-19-P3IA-0001 (PRAIRIE 3IA Institute) and ANR-19-P3IA-000 (ANITI 3IA Institute), Louis Vuitton ENS Chair on Artificial Intelligence, and the European project MEMMO (Grant 780684).

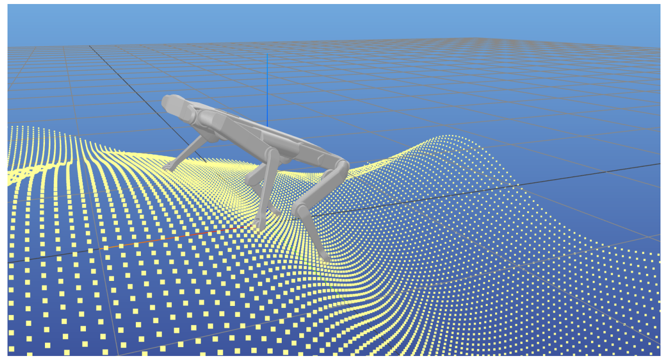

Fig. 5. Inverse geometry Solo-12 quadruped with a heightmap and accounting for contact forces.

References

[1] Wilson Jallet, Antoine Bambade, Nicolas Mansard, and Justin Carpentier. Constrained Differential Dynamic Programming: A primal-dual augmented Lagrangian approach. March 2022.

[2] Richard M. Murray, Zexiang Li, and S. Shankar Sastry. *A Mathematical Introduction to Robotic Manipulation*. CRC Press, first edition, December 2017.

[3] Magnus R. Hestenes. Multiplier and gradient methods. *Journal of Optimization Theory and Applications*, 4(5):303–320, November 1969.

[4] M. J. D. Powell. Algorithms for nonlinear constraints that use lagrangian functions. *Mathematical Programming*, 14(1):224–248, December 1978.

[5] R. T. Rockafellar. The multiplier method of Hestenes and Powell applied to convex programming. *Journal of Optimization Theory and Applications*, 12(6):555–562, December 1973.

[6] Philip E. Gill and Daniel P. Robinson. A primal-dual augmented Lagrangian. *Computational Optimization and Applications*, 51(1):1–25, January 2012.

[7] Taylor A. Howell, Brian E. Jackson, and Zachary Manchester. ALTRO: A Fast Solver for Constrained Trajectory Optimization. In *2019 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 7674–7679, Macau, China, November 2019. IEEE.

[8] Sarah Kazdadi, Justin Carpentier, and Jean Ponce. Equality Constrained Differential Dynamic Programming. In *ICRA 2021 - IEEE International Conference on Robotics and Automation*, May 2021.

[9] Wilson Jallet, Nicolas Mansard, and Justin Carpentier. Implicit Differential Dynamic Programming. In *International Conference on Robotics and Automation (ICRA 2022)*, Philadelphia, United States, May 2022. IEEE Robotics and Automation Society.

[10] Antoine Bambade, Sarah El-Kazdadi, Adrien Taylor, and Justin Carpentier. Prox-qp: Yet another quadratic programming solver for robotics and beyond. In *Robotics: Science and Systems 2022*, 2022.

[11] R. T. Rockafellar. Augmented Lagrangians and Applications of the Proximal Point Algorithm in Convex Programming. *Mathematics of Operations Research*, 1(2):97–116, 1976.

[12] Gaël Guennebaud, Benoît Jacob, et al. Eigen v3. http://eigen.tuxfamily.org, 2010.

[13] Justin Carpentier, Guilhem Saurel, Gabriele Buondonno, Joseph Mirabel, Florent Lamiraux, Olivier Stasse, and Nicolas Mansard. The Pinocchio C++ library – A fast and flexible implementation of rigid body dynamics algorithms and their analytical derivatives. In *IEEE International Symposium on System Integrations (SII)*, 2019.

[14] Justin Carpentier and Nicolas Mansard. Analytical Derivatives of Rigid Body Dynamics Algorithms. In *Robotics: Science and Systems XIV*. Robotics: Science and Systems Foundation, June 2018.

[15] Joel A. E. Andersson, Joris Gillis, Greg Horn, James B. Rawlings, and Moritz Diehl. CasADi: A software framework for nonlinear optimization and optimal control. *Mathematical Programming Computation*, 11(1):1–36, March 2019.