A Constrained Iterative LQR Solver for the Trajectory Optimization Framework Horizon

Arturo Laurenzi, Francesco Ruscelli, and Nikos G. Tsagarakis.

***Abstract*— This work introduces the implementation of the equality-constrained *Iterative Linear-Quadratic Regulator (eILQR)* solver used inside the recently published trajectory optimization (TO) framework for robotic application, *Horizon*. Differently from most constrained ILQR implementations that use an iterative penalty approach such as *Augmented Lagrangian*, we directly apply Newton's method to the KKT equations of the TO problem at hand, and show how to compute Newton steps with linear complexity w.r.t. the horizon length. Our approach generates estimates of Lagrangian multipliers corresponding to all constraints, allowing to use an exact ℓ_1 merit function inside a line-search based globalization strategy. We conclude our work with an extensive validation campaign involving many complex robotic platforms, such as our wheeled-legged quadruped robot CENTAURO.**

I. Introduction and Related Works

Trajectory optimization (TO) has recently attracted the interest of the robotics community thanks to its ability to generate complex, whole-body coordinated motions from an intuitive, high-level task description specified in terms of costs and constraints, also resulting in a plurality of available software tools. In most applications, trajectory optimization problems are natively written in continuous-time; it is however preferable[1] from a numerical point of view to first perform a discretization step that yields an ordinary non-linear program (NLP), such as the following:

$$\min_{x_{0:N},\, u_{0:N-1}} \quad \sum_{k=0}^{N-1} \ell_k(x_k,u_k) + \ell_N(x_N) \tag{1a}$$

$$\text{s.t.} \quad x_{k+1} = F(x_k,u_k) \tag{1b}$$

$$g_k(x_k,u_k) \geq 0, \quad g_N(x_N) \geq 0 \tag{1c}$$

$$h_k(x_k,u_k) = 0, \quad h_N(x_N) = 0, \tag{1d}$$

which can be handled via off-the-shelf NLP solvers. Such algorithms are applicable to generic NLP problems, but are likely to offer sub-optimal performance for any specific problem at hand. Indeed, TO problems in the form (1), exhibit a special structure that is due to the so-called Markov property, i.e. (i) cost and constraints are separable w.r.t. time, and (ii) the state vector at a next time $k+1$ can be expressed as a sole function of the state and input at the present time k. This key property is classically exploited by the well-known Iterative Linear Quadratic Regulator (ILQR) solver. Because of its parametric nature, extensions of ILQR including constraints have proven to be more challenging to devise. Approaches based on *Augmented Lagrangian* have been proposed [3], [4], which however can suffer from poor numerical conditioning due the possibly large required values of their penalty parameter, which moreover implies additional tunable knobs. Exact strategies for general eILQR were proposed e.g. in [5], which however includes a cubic complexity constraint handling routine. Recently, the methods of [6], [7] proposed a solution to eLQR that enjoys linear complexity w.r.t. N, limiting however to quadratic costs and linear constraints, and neglecting the computation of Lagrange multipliers.

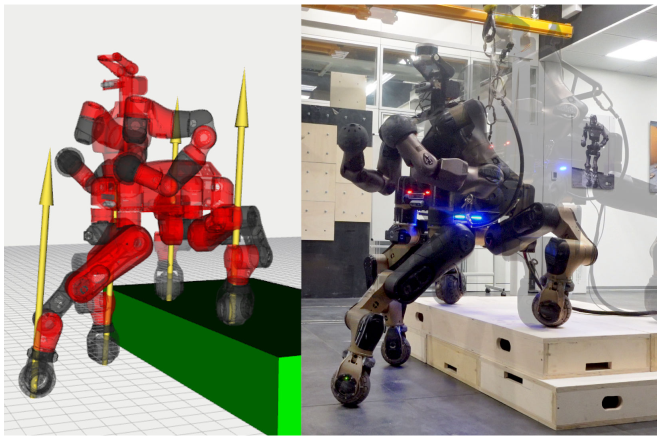
Fig. 1. The Centauro [2] robot climbing off a 30 cm platform thanks to the proposed eILQR solver. Visualization of optimized trajectory (left) and real robot execution (right).

This abstract introduces the implementation of an eILQR algorithm for our recently introduced TO framework, Horizon [8]. Our main contributions are the following:

- A derivation of eILQR that does not rely on Bellman's principle of optimality, and instead applies Newton's method to the first-order necessary condition of optimality for (1).
- Thanks to the proposed derivation, we compute values of the Lagrange multipliers, and show that they prove to be useful for globalization purposes, as well as to incorporate second-order information.
- Finally, we validate our eILQR algorithm with many TO instances involving quadruped and humanoid robots, ranging from offline-computed dynamic maneuvers, to online receding-horizon optimization.

Authors are with the Humanoid and Human-Centred Mechatronics (HHCM) Lab, Istituto Italiano di Tecnologia (IIT), Genova. Please send correspondence to {name.surname}@iit.it

[1]See [1, Sec. 4.3] for a discussion on this matter.

II. Proposed method

Since the TO (1) is an ordinary equality-constrained NLP, we can attempt to solve it by writing down the first-order necessary condition for the optimality of a candidate solution, i.e. the Karush-Kuhn-Tucker (KKT) conditions. To this aim let $\lambda_k \in \mathbb{R}^n$ and $\mu_k \in \mathbb{R}^{p_k}$, for $k \in \{0,\dots,N\}$, be the vectors of Lagrange multipliers corresponding to the dynamics constraint (1b)[2], and equality constraint (1d), respectively. The Lagrangian function for problem (1) is therefore defined as

$$\begin{aligned}&L(x_{0:N}, u_{0:N-1}, \lambda_{0:N-1}, \mu_{0:N}) = \\ &\sum_{k=0}^{N-1} \ell_k(x_k,u_k) + \mu_k^T h_k(x_k,u_k) + \lambda_k^T (F_k(x_k,u_k) - x_{k+1}) + \\ &+ \ell_N(x_N) + \mu_N^T h_N(x_N).\end{aligned} \tag{2}$$

The KKT conditions for (2) read as follows:

$$\frac{\partial L}{\partial x_k} = \nabla_x \ell_k + C_k^T \mu_k - \lambda_{k-1} + A_k^T \lambda_k = 0 \tag{3a}$$

$$\frac{\partial L}{\partial u_k} = \nabla_u \ell_k + D_k^T \mu_k + B_k^T \lambda_k = 0 \tag{3b}$$

$$\frac{\partial L}{\partial \lambda_k} = F_k(x_k,u_k) - x_{k+1} = 0 \tag{3c}$$

$$\frac{\partial L}{\partial \mu_k} = h_k(x_k,u_k) = 0, \tag{3d}$$

for $k \in \{0,\dots,N-1\}$, whereas for $k=N$ we have

$$\frac{\partial L}{\partial x_N} = \nabla_x \ell_N + C_N^T \mu_N - \lambda_{N-1} = 0 \tag{4a}$$

$$\frac{\partial L}{\partial \mu_k} = h_N(x_N) = 0. \tag{4b}$$

In (3) and (4) $\nabla_x \ell_k \in \mathbb{R}^n$ and $\nabla_u \ell_k \in \mathbb{R}^m$ are the gradients of the intermediate costs w.r.t. state and input, respectively; $A_k \in \mathbb{R}^{p_k \times n}$ and $B_k \in \mathbb{R}^{p_k \times m}$ are the Jacobian matrices of the dynamic constraint (1b) w.r.t. state and input, respectively; $C_k \in \mathbb{R}^{p_k \times n}$ and $D_k \in \mathbb{R}^{p_k \times m}$ are the Jacobian matrices of the equality constraint (1d) w.r.t. state and input, respectively; note that the dependency of all quantities upon the generic KKT candidate (x_k,u_k) had been omitted for the sake of brevity.

A. *Linearized KKT conditions*

In general (3) and (4) represent a system of non-linear equations, whose (possibly not unique) solution is a KKT point, i.e. a candidate to be a local minimizer of (1). Starting from an initial guess in terms of both state, input trajectories and Lagrange multipliers, we can apply Newton's method to iteratively compute increments of such quantities, i.e. δx_k, δu_k, $\delta \lambda_k$, and $\delta \mu_k$.

B. *Newton step computation: backward pass*

The linearized KKT block at the generic node k cannot be directly solved due to the dynamics-induced coupling with the previous and next nodes. On the other hand, solving the overall KKT system obtained by stacking them over he horizon length would violate the linear complexity requirement w.r.t. the horizon length N. We therefore take a Riccati approach, and use the linearized dynamics to propagate both cost and constraints backward in time, starting from the final node, and solving at each step for one single control input increment δu_k. We shall hypothesize that a relation of the form (5) holds for any $k+1 \in \{0,\dots,N\}$, i.e.

$$S_{k+1}\,\delta x_{k+1} + V_{k+1}^T\,\delta \nu_{k+1} - \delta\lambda_k = -s_{k+1} \tag{5a}$$

$$V_{k+1}\,\delta x_{k+1} = -v_{k+1}, \tag{5b}$$

for some appropriate choice of a symmetric positive-definite matrix $S_{k+1} \in \mathbb{R}^{n\times n}$, $V_{k+1} \in \mathbb{R}^{q_{k+1}\times n}$, $s_{k+1} \in \mathbb{R}^n$, and $v_{k+1} \in \mathbb{R}^{q_{k+1}}$; as it will become more clear towards the end of this section, (5b) accumulates all constraints that have not been satisfied by future control inputs δu_i, for $i > k$, together with their corresponding multipliers $\delta \nu_{k+1}$. Even after back-propagating such a relation via the dynamics, the resulting expression cannot be solved directly, as the resulting input-constraint matrix $\tilde{D}_k$ is in general rank-deficient. Intuitively this reflects the fact that, for any given state increment δx_k, a constraint at time k can not be fulfilled by the sole selection of an appropriate control δu_k: proper conditions must be satisfied by previous controls δu_i for $i<k$, too.

C. *Constraint back-propagation*

To compute δu_k, we (i) identify the part of the constraint that is *feasible* at time k, and (ii) define the remaining part to be the unsatisfied constraint at time k in (5b). This is accomplished via a QR decomposition of said constraint matrix. It is also necessary to remove Lagrangian multipliers corresponding to the infeasible constraint component, again exploiting the QR decomposition The reduced KKT system can finally be solved for both the control input increment, and the Lagrange multiplier increment of the feasible constraint component. As expected, the result is expressed in terms of a linear policy w.r.t. the unknown state increment, that we can compactly write as

$$\delta u_k = l_u^k + L_u^k\,\delta x_k \tag{6a}$$

$$\delta \eta_k = l_{\eta,k} + L_{\eta,k}\,\delta x_k, \tag{6b}$$

for

$$\begin{bmatrix} l_{u,k} & L_{u,k} \\ l_{\eta,k} & L_{\eta,k} \end{bmatrix} = -K_k^{-1} \begin{bmatrix} h_{u,k} & H_{ux,k} \\ w_k & W_k \end{bmatrix}. \tag{7}$$

D. *Newton step computation: initial state optimization*

Although most of the existing literature focuses on the case where the initial state x_0 is fixed, the ILQR algorithm poses no such restriction. Indeed, optimizing over the initial state finds applications in the field of offline trajectory optimization, where the initial system state is an actual design variable; notice also how in the context of equality-constrained ILQR, a fixed initial state is obtained with the simple linear constraint $x_0 = \bar{x}_0$.

Extending ILQR to initial state optimization is done by considering (5) evaluated at the initial node. Solving for the initial state increment δx_0 and constraint multiplier $\delta \nu_0$ completes the backward pass, and allows to initialize the forward recursion, as explained in the next subsection.

[2] λ_k is also known as the co-state vector.

E. Newton step computation: forward pass

The Newton step computation process is completed by providing recursive expressions for the state, dynamics multiplier, and constraint increments, starting from the known quantities δx_0 and δv_0.

F. Globalization

While the full Newton step $\delta X = (\delta x_{0:N}, \delta u_{0:N-1})$ enjoys local quadratic convergence close to a solution, it can perform poorly when the linear model of the KKT system is not accurate enough. Convergence to a local minimum can be promoted by enforcing the decrease of a suitable merit function via e.g. of the line-search procedure described in [9, Ch. 3], which scales down the computed step δX by a factor α until sufficient improvement is achieved. As discussed in [9], a merit function for a constrained problem should balance two conflicting objectives, i.e. feasibility on the one hand, and optimality on the other hand; furthermore, it is desirable for the chosen merit function to be *exact*, i.e. to possess a minimum point at any local solution of (1). One such function [9, Ch. 15.4] is the following exact ℓ_1 merit m:

$$m(X) = J(X) + \gamma \|C(X)\|_1, \tag{8}$$

where J and C are the total cost and constraint vector for (1), provided that the coefficient $\gamma > 0$ exceeds the maximum Lagrange multiplier at the solution, i.e.

$$\gamma > \max\{\|\lambda^*_{0:N-1}\|_\infty, \|\mu^*_{0:N}\|_\infty\}. \tag{9}$$

As too large values of γ are known to cause numerical issues, we conveniently exploit the computed multiplier estimates to compute a reasonable weight value.

III. Implementation and Validation

The proposed eILQR algorithm is implemented in C++, using Eigen3 for dense linear algebra, and made available via Python bindings to the recently published Horizon framework [8]. Within Horizon, we rely on CasADi [10] to act as a modeling language, as well as to compute all required derivatives with support to code generation, too. Multibody kinematics and dynamics are computed by the Pinocchio [11] library, including collision computations via HPP-FCL[3].

As our use case is robotics-driven, we validate eILQR on a vast collection of robotic examples, where it is employed as an offline TO solver as well as in an online, receding-horizon fashion according to a real-time iteration (RTI) scheme. The adopted dynamic model is a kino-dynamic one, similar to [12], [13], where the state is composed of the robot configuration q and its velocity v, whereas the accelerations $\dot{v}$ and contact forces F are taken as inputs; the resulting continuous-time dynamics is then integrated with a 4th-order Runge-Kutta scheme.

In all TO problems equality constraints play a dominant role, as they are adopted (i) to enforce the robot's centroidal dynamics (i.e., physical consistency between state and input), (ii) to model stance phase by imposing zero contact velocity, (iii) to impose a given clearance trajectory to swing feet, and (iv) to constrain (a sub-set of) the final configuration to a desired goal state, which is the main driver behind all the optimized motions. Costs are only used to regularize the solution, whereas unilateral contact forces, friction cones, and joint limits are modeled with soft barriers.

[3]The HPP-FCL library is available at `https://github.com/humanoid-path-planner/hpp-fcl`

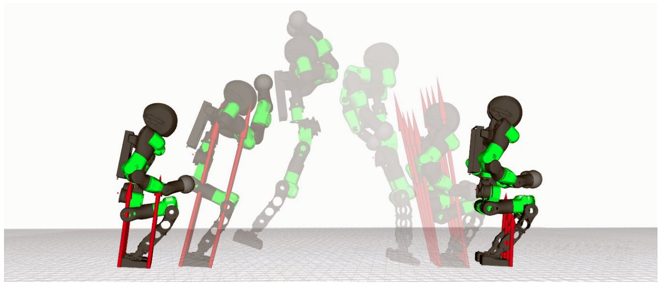

Fig. 2. Optimized trajectory requiring our humanoid robot COMAN+ to perform a dynamic 1.0m forward jump.

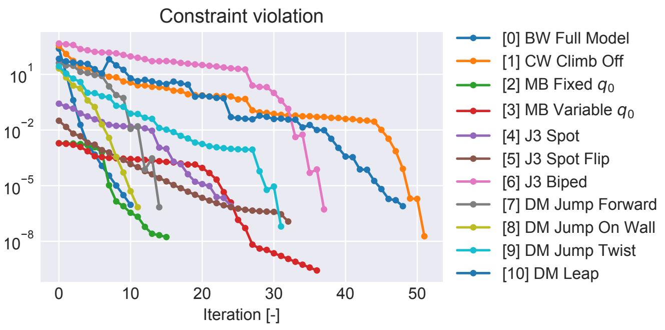

Fig. 3. History of constraint violations as the eILQR solver converges to a solution. The solver termination condition is set to $\varepsilon_{\text{feas}} = 10^{-6}$.

A. Discussion

As seen e.g. in Figure 3, eILQR can solve all problems within a modest number of iterations despite being seeded with a naive, constant initial guess. As shown in Figure 4, eILQR iterations often increase the cost value, highlighting the conflict between optimality and feasibility. The proposed merit function and regularization strategy (Figure 5) play

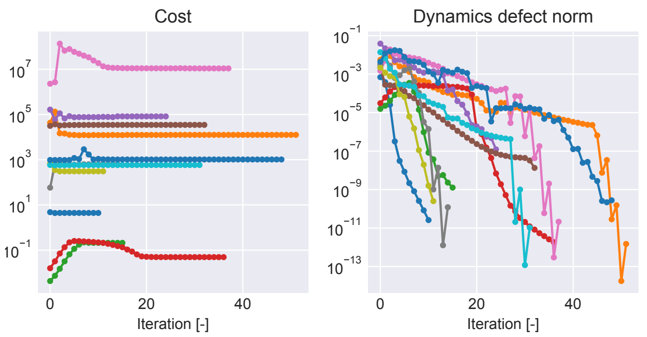

Fig. 4. Histories of cost values and dynamics defects (3c) as the eILQR solver converges to a solution.

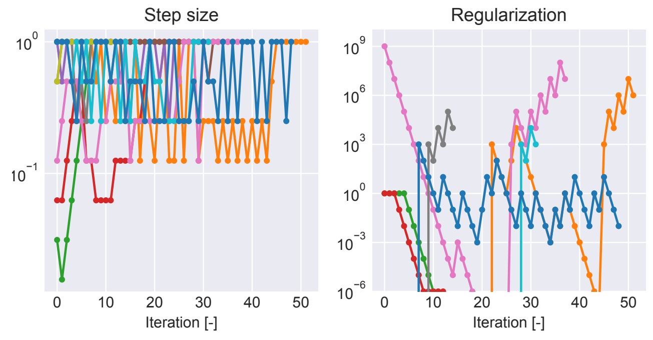

Fig. 5. Adaptive step size α and regularization ε_{reg} histories.

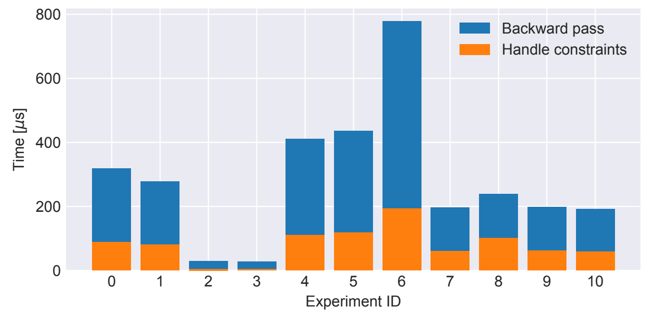

Fig. 6. Histogram of required CPU time to solve a single backward pass iteration (i.e., for a single node), as well as to perform the constraint backpropagation routine. Experiment IDs correspond to the legend of Figure 3.

therefore a crucial role in monitoring progress and result in a globally convergent algorithm. Finally, the computational impact of the constraint propagation routine of Section II-C on the backward pass is rather moderate, as seen in Figure 6.

IV. Conclusions

This abstract has introduced the implementation of the equality-constrained ILQR (eILQR) solver used inside the recently published Horizon framework [8] for trajectory optimization (TO). Future work will address the inclusion of inequality constraints, potentially exploiting active-set methods, which heavily rely on the provided Lagrange multiplier estimates.

Acknowledgments

The research leading to these results has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreements No. 101016007 (CONCERT) and No. 871237 (SOPHIA).

References

[1] J. T. Betts, *Practical Methods for Optimal Control and Estimation Using Nonlinear Programming, Second Edition*. Society for Industrial and Applied Mathematics, second ed., 2010.

[2] N. Kashiri, L. Baccelliere, L. Muratore, A. Laurenzi, Z. Ren, E. M. Hoffman, M. Kamedula, G. F. Rigano, J. Malzahn, S. Cordasco, *et al.*, "Centauro: A hybrid locomotion and high power resilient manipulation platform," *IEEE Robotics and Automation Letters*, vol. 4, no. 2, pp. 1595–1602, 2019.

[3] S. E. Kazdadi, J. Carpentier, and J. Ponce, "Equality constrained differential dynamic programming," in *2021 IEEE International Conference on Robotics and Automation (ICRA)*, pp. 8053–8059, 2021.

[4] T. A. Howell, B. E. Jackson, and Z. Manchester, "ALTRO: A Fast Solver for Constrained Trajectory Optimization," in *2019 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pp. 7674–7679, 2019.

[5] A. Sideris and L. A. Rodriguez, "A riccati approach to equality constrained linear quadratic optimal control," in *Proceedings of the 2010 American Control Conference*, pp. 5167–5172, 2010.

[6] F. Laine and C. Tomlin, "Efficient Computation of Feedback Control for Equality-Constrained LQR," in *2019 International Conference on Robotics and Automation (ICRA)*, pp. 6748–6754, 2019.

[7] S. Yang, G. Chen, Y. Zhang, F. Dellaert, and H. Choset, "Equality constrained linear optimal control with factor graphs," *2021 IEEE International Conference on Robotics and Automation (ICRA)*, pp. 9717–9723, 2021.

[8] F. Ruscelli, A. Laurenzi, N. G. Tsagarakis, and E. Mingo Hoffman, "Horizon: A trajectory optimization framework for robotic systems," *Frontiers in Robotics and AI*, vol. 9, 2022.

[9] J. Nocedal and S. J. Wright, *Numerical Optimization*. New York, NY, USA: Springer, 2e ed., 2006.

[10] J. A. E. Andersson, J. Gillis, G. Horn, J. B. Rawlings, and M. Diehl, "CasADi – A software framework for nonlinear optimization and optimal control," *Mathematical Programming Computation*, vol. 11, no. 1, pp. 1–36, 2019.

[11] J. Carpentier, G. Saurel, G. Buondonno, J. Mirabel, F. Lamiraux, O. Stasse, and N. 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.

[12] J.-P. Sleiman, F. Farshidian, M. V. Minniti, and M. Hutter, "A unified mpc framework for whole-body dynamic locomotion and manipulation," *IEEE Robotics and Automation Letters*, vol. 6, no. 3, pp. 4688–4695, 2021.

[13] H. Ferrolho, V. Ivan, W. Merkt, I. Havoutis, and S. Vijayakumar, "Inverse dynamics vs. forward dynamics in direct transcription formulations for trajectory optimization," in *2021 IEEE International Conference on Robotics and Automation (ICRA)*, pp. 12752–12758, 2021.