ICRA2026 Late Breaking Results Poster presented at
2026 IEEE International Conference on Robotics and Automation (ICRA 2026)
June 1-5, 2026. Vienna, Austria

Towards Massively Parallel Motion Planning with Inverse Dynamics

Ioannis Tsikelis[†] and Enrico Mingo Hoffman[†]

***Abstract*— Parallel evaluation of robotic system environments is becoming increasingly popular in modern robotics applications for machine learning and stochastic control. At the same time, the field of model-based control has matured enough to provide solutions that cover the needs of sophisticated robotics platforms. However, few works address the parallelization of such solvers to be combined with the above approaches and accelerate research in robot planning and control.**

We present preliminary results toward a novel implementation of a batched SQP solver for equality-constrained optimal control. After linearizing the dynamics in the SQP step, we employ a state-control equality constrained LQR solver. The additional equality constraints yield a structured system at each stage that can be solved via a Riccati-recursion-based block elimination.

We evaluate our approach on an inverse-dynamics-based optimal control problem, in contrast to the forward-dynamics formulations typical of related works. Our results demonstrate computational efficiency and structural advantages for massively parallel environments. Our implementation, available here, is developed in *PyTorch*, taking advantage of the library's batched linear algebra suite for parallelization.

I. Introduction

A Graphics Processing Unit (GPU) comprises many processors optimized for executing identical instructions in parallel. Despite their processors' limited capabilities, the use of GPUs has enabled great results in multiple robotics sub-fields [1], [2].

Model-based optimal control views the control problem of a system as an optimization task, minimizing an objective function over a time horizon subject to the system's dynamics. Its results are explainable and can guarantee safety and stability [3]. Massively parallel, also known as batched, model-based optimization software has shown great promise when combined with learning-based approaches [4], [5], or when utilized for the estimation and rejection of unknown disturbances [6].

However, the architectural limitations of GPUs pose a fundamental difficulty in implementing optimal control algorithms, leading to a visible contrast between the wide availability of optimal control solvers on the CPU and solvers that support multiple environments on the GPU and cover the diverse needs of roboticists.

In this work, we attempt to bridge the gap between the rich landscape of CPU optimal control software and its less diverse counterpart of GPU-based methods by proposing a non-linear optimal control solver based on Sequential Quadratic Programming (SQP), and is specifically designed for solving multiple optimal control instances in parallel. By utilizing the structure of the inner Quadratic Program (QP) solver, we are able to introduce inverse dynamics formulations without specific constraint handling approaches that may break synchronous execution.

Our experimental results demonstrate computational efficiency and structural advantages for massively parallel environments.

[†]Authors are with Inria, Université de Lorraine, CNRS, 54000 Nancy, France. ioannis.tsikelis@inria.fr, enrico.mingo-hoffman@inria.fr

This work was created with the co-financing of the French National Research Agency (ANR) under the project ANR-24-CE33-0753-01 (MeRLin).

II. Methodology

In nonlinear optimal control, we are concerned with numerical optimization problems where arbitrary costs and constraints can be defined at every stage. Each stage is connected to the next one by a discretized dynamics constraint. In this work, we consider a nonlinear optimal control problem in the following form:

$$\begin{aligned} \min_{\mathbf{x}_k, \mathbf{u}_k} \quad & \ell_N(\mathbf{x}_N) + \sum_{k=0}^{N-1} \ell_k(\mathbf{x}_k, \mathbf{u}_k) \\ \text{s.t.} \quad & \mathbf{x}_{k+1} = \mathbf{f}(\mathbf{x}_k, \mathbf{u}_k), \quad k = 0, \dots, N-1, \\ & \mathbf{h}_k(\mathbf{x}_k, \mathbf{u}_k) = \mathbf{0}, \quad k = 0, \dots, N-1, \end{aligned} \tag{1}$$

where the decision variables are partitioned to $N+1$ *states* $\mathbf{x} \in \mathbb{R}^{n_x}$ and N *controls* $\mathbf{u} \in \mathbb{R}^{n_u}$, $\mathbf{f}(\mathbf{x}_k, \mathbf{u}_k) \in \mathbb{R}^{n_x}$ is the discretized dynamics (or integration) function, $\ell_k(\mathbf{x}_k, \mathbf{u}_k)$ are scalar *running* costs, $\ell_N(\mathbf{x}_N)$ a scalar *final* cost, and $\mathbf{h}_k(\mathbf{x}_k, \mathbf{u}_k) = \mathbf{0} \in \mathbb{R}^h$ stage-wise constraints involving both states and controls. The N state-action pairs (also known as *stages* or *nodes*) define the *prediction horizon* of the optimal control problem.

A. SQP With Equality Constrained LQR

The SQP approach iteratively considers the Taylor expansion of the Lagrangian optimal control problem of Eq. (1), around the current solution iterate. The linearization describes a QP with a block-diagonal structure, which allows the use of a block elimination algorithm (also known as the *Riccati recursion*) to efficiently compute the optimal values of the corrections to the solution variables and the associated Lagrange multipliers.

Given that the additional equality constraints concern both the state and control variables (as is the case in inverse dynamics formulations), we can augment each control variable $\mathbf{u}_k$ with the Lagrange multipliers of the additional stage-wise constraints. By also augmenting the related linearization matrices with the constraint Jacobians, we can make use of the Riccati recursion algorithm to solve the inner QP respecting the additional equality constraints.

TABLE I

GPU NATIVE NON-LINEAR SOLVERS

	NLP Method	QP Solver	Gen. Constraints	Real-time	Derivatives	Scalable
SQP-EQ (Ours)	SQP	LQR-EQ	Yes (Equality)	No	Analytical/Auto-diff	Yes
diffmpc	SQP	PCG	No	Yes	Auto-diff	No
GATO	SQP	PCG	No	Yes	Analytical	Yes

B. Line Search And Termination

For the line search step of the SQP algorithm, we use a filter line search approach. The line search procedure is not guaranteed to finish at the same time for all iterates. We then employ a simple termination criterion where we terminate the optimization when both the square norm of the QP solver corrections and the constraint violations are below a certain threshold or until a pre-determined amount of iterations is reached.

Because of the need for parallel execution of the operations of each environment, we use boolean update and termination masks for the `for` loops of the line search and termination procedures [1].

III. EXPERIMENTS

We evaluate our solver focusing on the speedup capabilities(Fig. 1), efficiency of the inner QP solver(Tab. II) with other approaches and overall solver performance with the state of the art(Tab. I). For our experiments, we use the textbook example of the cart-pole system [7].

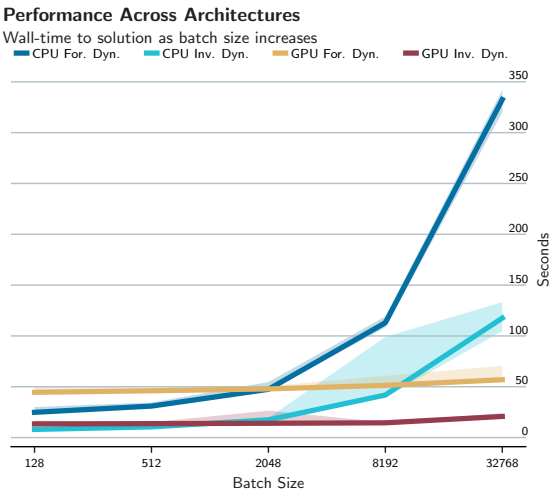

Fig. 1. Wall-time in seconds per environments solved in parallel. Initial state injected with Gaussian noise of $\mu = 0, \sigma = 0.01$. Solid lines indicate the median over 10 iterations of the experiment, and the shaded regions are the regions between the 5th and 95th percentiles.

IV. CONCLUSIONS

We presented a memory-efficient SQP optimal control solver that supports arbitrary state-control equality constraints, permitting the use of inverse dynamics formulations in the optimization problem. We evaluated our proposition on a toy underactuated dynamics model, providing both qualitative and quantitative comparisons with the current state of the art. Our results show a promising direction for optimal control software to accelerate the fields of stochastic optimal control and robot learning.

Future work will focus on improving the current approach with more complex underactuated systems. The current line search and termination check methods can also be improved upon.

Additional pursuits include support for inequality constraints and differentiability support to allow the use of the SQP optimization step as a neural network layer. Finally, we plan on an optimized version of our solver, that could also include GPU accelerations for each environment within the batched solver and could ideally run real-time for *model predictive control* (MPC) applications.

TABLE II

INNER QP SOLVER COMPARISON:

Median, 5th, and 95th percentiles of solve time across all QP solves during execution, Mbytes of VRAM allocated per environment, and maximum violation of the dynamics and under-actuation constraints. The experiment was repeated 10 times with additive noise on the initial state.

QP Solver	**5%t**	**50%t**	**95%t**	**MBytes/Env**	$\|viol_{dyn}\|_\infty$	$\|viol_{uact}\|_\infty$
Forward dynamics						
LQR	0.41s	0.42s	0.57s	**2.88**	9.73e-05	–
LU (dense)	**0.37**s	**0.38s**	**0.39s**	15.73	9.92e-05	–
QPTH	0.47s	0.54s	0.69s	20.45	9.85e-05	–
QPTH-IR	1.12s	1.13s	1.27s	27.00	1.00e-04	–
Inverse dynamics						
LQR-EQ	0.24s	0.25s	0.32s	**2.88**	9.54e-07	9.99e-05
LU (dense)	**0.19s**	**0.20s**	**0.25s**	20.97	9.54e-07	1.00e-04
QPTH	0.30s	0.38s	0.45s	31.20	3.55e-10	1.00e-04
QPTH-IR	1.22s	1.24s	1.30s	45.09	1.65e-14	1.00e-04

REFERENCES

[1] N. Rudin et al., "Learning to walk in minutes using massively parallel deep reinforcement learning," in *Proceedings of the 5th Conference on Robot Learning*, ser. Proceedings of Machine Learning Research, vol. 164, PMLR, 2022, pp. 91–100.

[2] B. Vlahov et al., *Mppi-generic: A cuda library for stochastic trajectory optimization*, 2024. arXiv: 2409.07563.

[3] R. Grandia et al., "Perceptive locomotion through nonlinear model-predictive control," *IEEE Transactions on Robotics*, vol. 39, no. 5, pp. 3402–3421, 2023.

[4] S. H. Jeon et al., *Residual mpc: Blending reinforcement learning with gpu-parallelized model predictive control*, 2025. eprint: arXiv:2510.12717.

[5] B. Amos et al., "Differentiable MPC for End-to-end Planning and Control," 2018.

[6] A. Du et al., *Gato: Gpu-accelerated and batched trajectory optimization for scalable edge model predictive control*, 2025.

[7] R. Tedrake, *Underactuated Robotics, Algorithms for Walking, Running, Swimming, Flying, and Manipulation*. 2023. [Online]. Available: https://underactuated.csail.mit.edu.