Benchmarking Different QP Formulations and Solvers for Dynamic Quadrupedal Walking

Franek Stark[1] and Jakob Middelberg[1,2] and Dennis Mronga[1] and Shubham Vyas[1,2] and Frank Kirchner[1,2]

***Abstract*— Quadratic Programs (QPs) are widely used in the control of walking robots, especially in Model Predictive Control (MPC) and Whole-Body Control (WBC). In both cases, the controller design requires the formulation of a QP and the selection of a suitable QP solver, both requiring considerable time and expertise. While computational performance benchmarks exist for QP solvers, studies comparing optimal combinations of computational hardware (HW), QP formulation, and solver performance are lacking. In this work, we compare dense and sparse QP formulations, and multiple solving methods on different HW architectures, focusing on their computational efficiency in dynamic walking of four-legged robots using MPC. We introduce the Solve Frequency per Watt (SFPW) as a performance measure to enable a cross-hardware comparison of the efficiency of QP solvers. We also benchmark different QP solvers for WBC that we use for trajectory stabilization in quadrupedal walking. As a result, this paper provides recommendations for the selection of QP formulations and solvers for different HW architectures in walking robots and indicates which problems should be devoted the greater technical effort in this domain in future.**

I. INTRODUCTION

The development of quadrupedal robots has progressed rapidly in recent years, and several platforms have now reached a level of industrial maturity [1], [2], [3]. Apart from advancements in actuation, the main progress has been made on numerical methods for trajectory optimization, and its online implementation Model Predictive Control (MPC). At the ICRA 2022 conference, 68 % of the papers presented at the legged robotics workshop covered receding horizon control or MPC [4]. Today, the de facto standard approach for legged locomotion comprises the planning of a contact sequence, computing a corresponding Center of Mass (COM) trajectory using MPC, and, finally, stabilizing the obtained motion in real-time using Whole-Body Control (WBC) [5].

At the core of MPC, an optimal control problem has to be solved, which is formulated as linearly constrained Quadratic Program (QP). Solving constrained QPs is such a fundamental problem in both, MPC and WBC, that a huge amount of research has been devoted to developing efficient and stable QP solvers, most of which are based on the Active-set method (ASM), Interior-point method (IPM) or Augmented Lagrangian Method (ALM). Each solver and each method is best suited for specific applications, e.g., some solvers show their strengths with small QPs on embedded systems [7], others are optimized for larger problems that may arise in machine learning applications [8], [9], and again others exploit the sparsity in the QP [10], [8], a feature that can be advantageous in MPC applications. Here, sparsity in the QP is characteristic due to diagonal stacking of system state and control matrices over the entire prediction horizon. To reduce the QP size, the state variables can be eliminated as decision variables, leading to denser and smaller matrices [11], a process that is known as *condensing*. A great variety of problem formulations can be obtained by partially condensing the QP, e.g., only eliminating half of the state variables [12]. Special forms of condensed QPs can be found in [11] and [13]. In the works that introduce the current de facto standard of using MPC and WBC [13], [14], the authors claim significant speed-up by removing some constraints and state variables from the QP, leading to a dense and compact formulation. However, according to [12], sparse formulations are advantageous in MPC as their computational and memory requirements grow linearly with the prediction horizon ($\mathcal{O}(N)$) if sparsity is exploited, while for the dense variant, the computational cost is $\mathcal{O}(N^3)$ for IPM and $\mathcal{O}(N^2)$ for ASM [15]. Given these seemingly contradicting results, the effects of partial or full condensing of the MPC problem with respect to the solver need to be investigated more thoroughly.

There are several open-source benchmarks of QP solvers [16], [17], which mostly use accuracy and solution time as a benchmark. However, as legged systems are being

Fig. 1. Go2 quadruped [6] used for evaluation in a Volcanic field test in an analogous scenario with limited onboard computing power.

This work was done in the AAPLE (grant number 50WK2275) and M-Rock (grant number 01IW21002) projects funded by the German Federal Ministry for Economic Affairs and Climate Action (BMWK) and the Ministry of Education and Research (BMBF) and is supported with funds from the federal state of Bremen for setting up the Underactuated Robotics Lab (201-342-04-2/2024-4-1)

Special thanks goes to Hannah Isermann and Rohit Kumar for their support in developing control software for quadrupedal walking.

[1]All authors are with the Robotics Innovation Center at the German Research Center for Artificial Intelligence (DFKI), Bremen, Germany, Corresponding author's email: franek.stark@dfki.de

[2]Jakob Middelberg, Shubham Vyas, and Frank Kirchner are additionally affiliated with the University of Bremen, Bremen, Germany

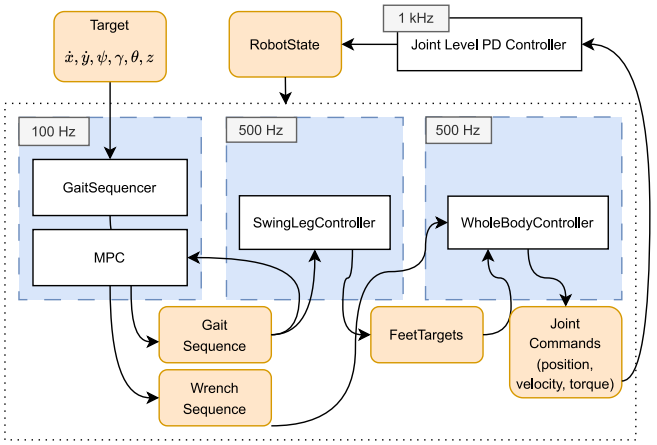

Fig. 2. Block diagram showing dynamic walking controller's architecture

proposed for tasks such as space exploration [18], [19], other metrics such as power consumption and required onboard computing power become a critical factor for long-duration autonomous missions. Fig. 1 illustrates the quadruped robot during a space exploration field test on a volcano, where it operates on limited battery capacity and onboard computing power to carry out the exploration tasks. Despite the importance of energy constraints, the impact of QP formulation (sparse, partially condensed, condensed) and the relation to the prediction horizon, solver, and the computing hardware (HW) on performance used has yet to be comprehensively examined.

In this work, we focus on the application of dynamic quadrupedal walking using MPC, which has so far produced several specialized methods for QP formulation and solving, to reduce computation time or increase the planning horizon [20], [13], [14]. We employ the standard approach to quadrupedal walking (contact planning, MPC, WBC) and evaluate it on a Unitree Go2 robot [6]. The benchmark involves (1) different QP formulations in MPC (sparse, partially condensed, and fully condensed), (2) two HW architectures (x86, ARM) with desktop and single-board target computers, (3) various QP solvers, including different principled methods for convex optimization, (4) different (dense) QP solvers for WBC. To allow a cross-HW assessment of solver efficiency, we introduce the *Solve Frequency per Watt* (SFPW) metric to compare different solvers. As a result of our benchmark, we recommend optimal combinations of computing HW, problem sparsity, and QP solvers for dynamic legged locomotion. The entire benchmark code, including the quadruped controller, is made open source[1].

The remaining paper is structured as follows. Section II describes the dynamic walking controller, Section III the experimental setup and the performance metrics used for benchmarking. Section IV summarizes the results, with discussion in section V, and Section VI draws conclusions on the benchmark.

[1]https://github.com/dfki-ric-underactuated-lab/dfki-quad

II. DYNAMIC WALKING CONTROLLER

Our dynamic walking controller follows the general approach as described by [13] and [14] and is illustrated in Fig. 2. It outputs joint commands (position, velocity, torque) based on a target linear and rotational walking velocity and body posture. The four sub-components of the controller run in parallel at different control rates: (1) The *Gait Sequencer (GS)* heuristically determines a gait sequence, containing the foot contact plan and the robot's target poses and velocities for the respective time steps over the prediction horizon N, based on the selected gait type and control target. (2) The *MPC* calculates the optimal contact forces for the feet over the prediction horizon to reach the target pose and velocity specified in the gait sequence, assuming single rigid-body dynamics (SRBD). (3) For the feet not intended for ground contact in the current time step, the Swing Leg Controller (SLC) computes the trajectory using a Bézier curve. (4) The *WBC* computes the optimal joint accelerations and torques, which achieve the given target positions and velocities for the feet, the body posture/velocity and contact forces, taking into account the whole-body dynamics. GS and MPC run at 100 Hz, while SLC and WBC run at 500 Hz. The largest computing load is generated by MPC and WBC, which both require solving QPs in real-time. These two components are detailed in the following.

A. Model Predictive Control

1) Formulation: The MPC with N prediction steps is formulated as the following convex optimization problem:

$$\min_{\mathbb{X},\mathbb{U}} \quad \sum_{k=1}^{N} (\mathbf{x}_k - \mathbf{x}_k^d)^T \mathbf{Q}\ (\mathbf{x}_k - \mathbf{x}_k^d) + \mathbf{u}_{k-1}^T\ \mathbf{R}\ \mathbf{u}_{k-1} \tag{1a}$$

$$\text{s.t.} \quad \mathbf{x}_{k+1} = \mathbf{A}\mathbf{x}_k + \mathbf{B}_k\mathbf{u}_k, \qquad k = 0 \ldots N-1 \tag{1b}$$

$$\mathbf{f}_{m,k}^j \leq \mathbf{f}_k^j \leq \mathbf{f}_{M,k}^j, \qquad \forall j,\ k = 0 \ldots N-1 \tag{1c}$$

$$|\mathbf{f}_k^j \cdot \hat{\mathbf{e}}_n| \leq \mu \mathbf{f}_k^j \cdot \hat{\mathbf{e}}_z, \quad \forall n \in \{x, y\}, \forall j,\ k = 0 \ldots N-1 \tag{1d}$$

$$\mathbf{x}_0 = \bar{\mathbf{x}} \tag{1e}$$

The decision variables $\mathbf{x}_k \in \mathbb{X}, \mathbf{u}_k \in \mathbb{U}$, denote the prediction of the state and control input k steps ahead at the current time step, $\mathbf{Q}$ is the diagonal state error cost matrix, $\mathbf{R}$ is the diagonal input cost matrix, and $\mathbf{x}_k^d$ the target state from GS. The equality constraint (1b) formulates the linearized system dynamics in world coordinates. The state vector $\mathbf{x} \in \mathbb{R}^{13}$ contains the system's base orientation, position, linear, angular velocity, and the gravity constant. The control input $\mathbf{u} \in \mathbb{R}^{12}$ is a vector containing the four leg input forces $\mathbf{f}^j \in \mathbb{R}^3$, where $j \in \{1,2,3,4\}$ is the leg index. The state matrix $\mathbf{A}$ is linearized around the current state $\bar{\mathbf{x}}$. The input matrix $\mathbf{B}_k \in \mathbb{R}^{13\times12}$ consists of four $\mathbb{R}^{13\times3}$ blocks, mapping the respective feet contact force onto the system's state, depending on the planned contacts. The bounding-box constraint (1c) limits the contact force to a maximum $\mathbf{f}_{\mathrm{M},k}^j$, enforces a minimum contact force $\mathbf{f}_{\mathrm{m},k}^j$ and sets it to zero if the respective leg j is scheduled for swing phase at predicted step k. The constraint (1d) keeps the contact forces within a linearized friction cone

with coefficient μ, where $\hat{\mathbf{e}}_x, \hat{\mathbf{e}}_y$, and $\hat{\mathbf{e}}_z$ denote the standard unit vectors. The equality constraint (1e) sets the initial state to the current estimated state $\bar{\mathbf{x}}$.

2) Partial Condensing: Classically, MPC problems are formulated into a *dense* QP by eliminating all state variables (except the initial state) as the state can be expressed as a function of the previous state and the input in LTI systems. This work follows the idea of [12], implemented by [21], where the number of eliminated state variables can be specified. For this purpose, the original prediction horizon N is divided into N_p blocks, each containing several consecutive states. The idea is that in each block, all states but the first one are eliminated by the system dynamics. Following this approach, a partially condensed QP is formulated, which can be interpreted as another MPC problem with a prediction horizon of N_p, enlarged input matrix, and input vector, but fewer states and dynamic constraints. Further details can be found in [21].

B. Whole-Body Control

We use a variant of the WBC in [22] to stabilize the quadrupedal walking. The WBC considers the full system dynamics and solves for the joint accelerations $\ddot{\mathbf{q}} \in \mathbb{R}^{18}$ and contact forces $\mathbf{u} \in \mathbb{R}^{12}$ in a single QP:

$$\min_{\ddot{\mathbf{q}},\mathbf{u}} \quad \|\sum_i w_i(\mathbf{J}^i\ddot{\mathbf{q}} + \dot{\mathbf{J}}^i\dot{\mathbf{q}} - \dot{\mathbf{v}}_d^i)\|_2 + \|\mathbf{w}_f(\mathbf{u}_d - \mathbf{u})\|_2 \tag{2a}$$

$$\text{s.t.} \quad \mathbf{H}\ddot{\mathbf{q}} + \mathbf{h} = \mathbf{J}_c^T\mathbf{u} \tag{2b}$$

$$\mathbf{J}_c\ddot{\mathbf{q}} = -\dot{\mathbf{J}}_c\dot{\mathbf{q}} \tag{2c}$$

$$|\mathbf{f}^j \cdot \hat{\mathbf{e}}_n| \leq \mu \mathbf{f}^j \cdot \hat{\mathbf{e}}_z, \forall n \in \{x,y\}, \mathbf{f}^j \cdot \hat{\mathbf{e}}_z > 0, \ \forall j \tag{2d}$$

$$\tau_m \leq \mathbf{S}^{-1}\left(\mathbf{H}\ddot{\mathbf{q}} + \mathbf{h} - \mathbf{J}_c^T\mathbf{u}\right) \leq \tau_M \tag{2e}$$

where w_i is the weight, $\mathbf{J}^i \in \mathbb{R}^{6\times 18}$ the Jacobian, and $\dot{\mathbf{v}}_d^i \in \mathbb{R}^6$ the desired spatial acceleration for the i-th positioning task. The $\dot{\mathbf{v}}_d^i$ for the respective tasks are generated by PD-controllers to stabilize the body and feet trajectories produced by the MPC and SLC. Further $\mathbf{u}_d \in \mathbb{R}^{12}$ the desired contact forces, as optimized by the MPC in (1), i.e. $\mathbf{u}_d = \mathbf{u}_{k=0}$. The WBC can correct it to account for the full system dynamic. The amount of correction is specified by the diagonal weight matrix $\mathbf{w}_f$. The constraint (2b) considers the rigid body dynamics, where $\mathbf{H} \in \mathbb{R}^{18\times 18}, \mathbf{h} \in \mathbb{R}^{18}, \mathbf{J}_c \in \mathbb{R}^{12\times 18}$ are the mass-inertia matrix, bias vector, and contact Jacobian. Note that we consider only the floating base dynamics here , as we do not have torques as optimization variables in the QP. The constraint (2c) ensures that the feet' contacts are rigid and slip-free, while (2d) prevents the contact forces from leaving the linearized friction cones, where μ is the friction coefficient. Finally, (2e) ensures that the torque limits $[\tau_m, \tau_M]$ are respected, where $\mathbf{S} \in \mathbb{R}^{18\times 12}$ is the actuator selection matrix. Note that in (2e) we use the subscript a to account only for the actuated joints. In contrast to [14], we use a weighting scheme for prioritization, as it is the common choice in WBC and is thus better for benchmarking. The resulting joint accelerations $\ddot{\mathbf{q}}$ are integrated twice and the joint torques are computed from $\mathbf{q}, \dot{\mathbf{q}}, \ddot{\mathbf{q}}$ by inverse dynamics.

TABLE I
TARGET COMPUTERS USED FOR COMPARISON.

	Arch.	*CPU*	*Cores*	*RAM*
Jetson Orin NX	ARM64	AA78AEv8.2	8@2 GHz	16 GB
LattePanda Alpha	x86-64	M3-8100Y	2@3.4 GHz	8 GB
Desktop PC	x86-64	i9-10900K	10@3.7 GHz	16 GB

As alternative approach, we formulate the joint torques as additional optimization variables and solve for them directly in the QP. This leads to a larger QP, but avoids additional inverse dynamics computations. We refer to this approach as *full TSID* here, while (2) is called *reduced TSID*.

III. EXPERIMENTAL SETUP

A. Performance Metrics

Two different metrics are used to compare the performance of the QP solvers: (1) The solve time is the duration required for the solvers to solve the respective QP. (2) The *Solve Frequency per Watt (SFPW)*, which we introduce to assess the efficiency of the respective QP solvers independent of the computational HW. This metric is inspired by the *FLOPS per Watt* measure, employed by the Green500 list of the world's most power-efficient supercomputers [23]:

$$\text{SFPW} = \frac{\text{solve time}^{-1}}{\text{CPU power consumption}} \quad \left[\frac{Hz}{W}\right] \tag{3}$$

On x86, the Intel `RAPL` Interface determines the CPU power consumption. For the ARM, the `tegrastats` utility is used. The CPU power consumption is sampled at 10 Hz.

The three target computers are listed in Table I. The Jetson Orin NX and LattePanda Alpha are single-board computers suitable for installation on the quadruped due to their low power consumption and small form factor.

B. Implementation and Solvers

We use the *Unitree Go2* quadruped [6] for experimental evaluation. It is simulated using the Drake toolbox [24]. The controller is implemented using ROS 2.

The MPC is implemented using the QP interface of the acados framework [25]. It interfaces to a set of state-of-the-art QP solvers, including HPIPM [10], which provides the (partial) condensing routines from [21]. The WBC is implemented in the ARC-OPT framework [26], [27], which also comes with a set of QP solvers and different WBC implementations, including the *full* and *reduced TSID* described in Section II-B. Table II lists all solvers that are considered in this work. A sparse interface means that a solver uses the sparse MPC formulation as input and potentially exploits these. In addition, the degree of sparsity of these solvers can be controlled via the partial condensation routines. Some solvers only accept the fully dense MPC formulation as input, which means that comparison in terms of sparsity is not possible. We recognize that adjusting the hyperparameters, albeit time-consuming, can further improve performance.

TABLE II
QP SOLVERS COMPARED IN THIS WORK

Solver	*Method*	*Interface*	*MPC*	*WBC*
HPIPM [10]	IPM	Sparse, Dense	✓	✓
OSQP [8]	ADMM*	Sparse	✓	
qpOASES [7]	parametric ASM	Dense	✓	✓
DAQP [28]	dual ASM	Dense	✓	
Eiquadproq [29]	dual ASM†	Dense		✓
PROXQP [30]	ALM	Dense		✓

*Variant of ALM, †Algorithm of Goldfarb and Idnani [31]

However, all solvers are used with the standard hyperparameters to ensure better comparability. The qpOASES solver is used with the *MPC* option set. The HPIPM solver has predefined modes that adapt the underlying IPM algorithm. Here, the modes *balanced* and *speed_abs* are used and treated as two different solvers. While the first mode provides more accurate results, the second focuses on speed [10], which is more suitable for smaller systems and is equivalent to the HPMPC solver [32]. All solvers are warm-started if possible.

C. Problem Sizes

For comparison, we choose a short ($N = 10$) and medium ($N = 20$) MPC prediction horizon. The QP dimension for both MPC and WBC are shown in Table III. For the MPC solvers with sparse interfaces, this work compares all condensing levels such that the prediction horizon of the condensed QPs are $N_p \in \{N, N-1, \ldots, 1\}$. The prediction horizon of the original MPC is thereby evenly distributed into different blocks of size $\lfloor \frac{N}{N_p} \rfloor$. If N is not an integer multiple of N_p, the remainder of $\frac{N}{N_p}$ is distributed to the foremost blocks (one per block). The final state is always left out during partial condensing and stays at size 1. For the MPC solvers with the dense interface, the QP is fully condensed (including the terminal state), which is equivalent to partial condensing with $N_p = 0$. Note that the size of the constraint matrices in WBC will change dynamically when changing the contacts during walking.

TABLE III
QP PROBLEM SIZES

			Decision-variables	*Constraints*	
				Equality	*Inequality*
WBC	*Red. TSID*		30	18	28
WBC	*Full TSID*		42	30	28
MPC	*original*	$N = 10$	263	143	280
MPC	*original*	$N = 20$	513	273	360
MPC	*condensed*		$13N_p + 12N + 13$	$13N_p + 13$	$28N$

D. Test Scenarios

Two distinct scenarios are evaluated for each test case. In the first scenario, the controller performs a trotting gait, executing various velocity profiles with a total duration of $\approx$40 s. This induces dynamic motions with speeds up to $0.5\,\mathrm{m\,s^{-1}}$ and rotational velocities up to $40\,^\circ\mathrm{s^{-1}}$. In the second scenario the quadruped remains in standing mode, while responding to commanded body roll, pitch, and yaw movements up to $\pm 30^\circ$, as well as height adjustments of ± 10 cm. Here, the total duration is $\approx$20 s. If the robot falls, the attempt is repeated twice before the test case is counted as a *failure*. The dynamic simulation is run on a different computer in real time as to not affect the measurements.

IV. EXPERIMENTAL RESULTS

A. MPC Solve Time Analysis

The mean MPC solve time for the different solvers, target platforms and condensing levels for the *trotting* experiment is depicted in Fig. 3. The plotted time includes the condensing time, which is less than 1 ms. Overall, solve time increases with increasing density of the QP. For $N = 20$, the dense formulation takes too long to solve for certain solvers leading to failed experiments, as indicated by red circles in Fig. 3. For the sparse solvers that support different levels of condensing (HPIPM, OSQP), for $N_p \leq \frac{N}{2}$ the solve time increases exponentially with higher density, while it is approximately constant for $N_p > \frac{N}{2}$. The dense solvers (qpOASES, DAQP) perform better for fully condensed QPs, but are outperformed by solvers that exploit sparsity. As expected, all solvers show lower solve times for $N = 10$ than for $N = 20$. For all computer architectures, condensing levels, and horizons, the best-performing solver is HPIPM speed_abs. For $N = 20$, the second-fastest solver is OSQP, followed by HPIPM balance, although the difference is marginal on x86 systems. DAQP and qpOASES are much slower for this prediction horizon. For $N = 10$, however, both qpOASES and DAQP perform almost as well as OSQP with $N_p = 20$ (sparse problem). The x86 desktop architecture is the fastest among the target systems compared, with HPIPM speed_abs achieving sub-ms solve times. The Jetson and the LattePanda perform similarly, with the LattePanda being only slightly faster. To compare the MPC solve time between different test scenarios (standing, trotting), we use the normalized solve time histograms for all solvers on all computer architectures (see Fig. 4). The histograms show that the mean solve time is less for the standing scenario than for trotting on all solvers. The most significant differences can be seen in DAQP, OSQP and qpOASES, where the histogram peaks are separated from each other. This effect is most prominent for qpOASES. For the individual scenarios in themselves, the solution times for HPIPM are approximately normally distributed, while they show two significant peaks for the other solvers in trotting. When comparing target computers, the slowest system (LattePanda) shows the highest variance in solve times.

B. WBC Solve Time Analysis

Fig. 5 shows the total computation time for one WBC cycle (set up time for the QP + solve time) using different solvers and two different WBC types. The set up time

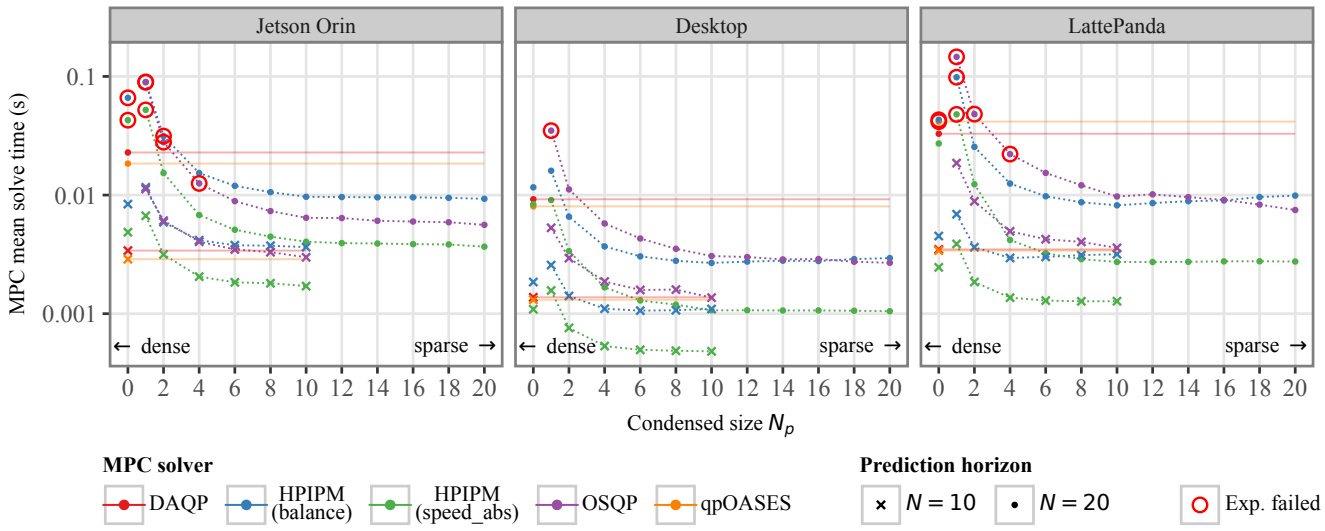

Fig. 3. Mean MPC solution time for all solvers, target platforms and condensation levels for $N = 20$ and $N = 10$ in the trotting experiment. The qpOASES and DAQP solvers only show single points, as no comparison of condensation levels is possible here. Note that higher N_p means the QP is sparser.

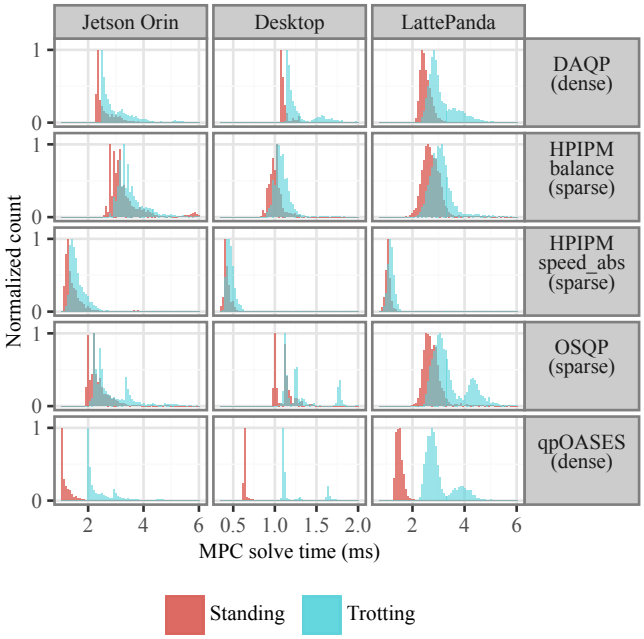

Fig. 4. Histogram of the MPC ($N = 10$) solve times for selected solvers on all target platforms for dynamic trotting compared to standing.

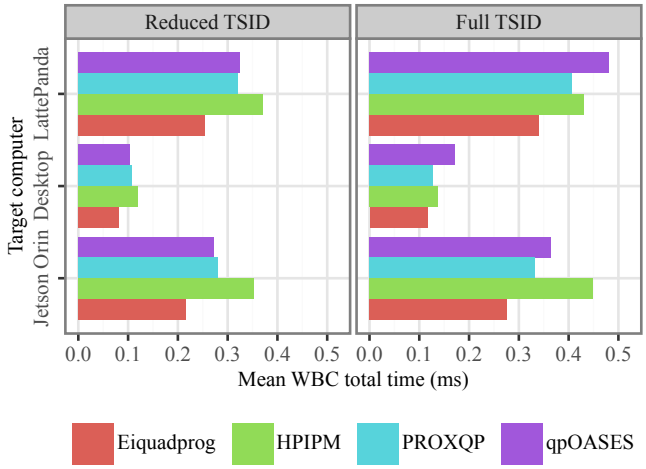

Fig. 5. Computation time for different solvers using the *reduced* and *full TSID* WBC in the trotting scenario.

for the QP is solver-independent and thus not considered here separately. The plots show that the Eiquadprog solver performs best in all comparisons. However, all solvers show a rather low computation time with $< 0.5ms$ on average. The computation time is slightly larger for the *full TSID*, as compared to the *reduced TSID* formulation on average.

C. Efficiency Analysis

Table IV shows the efficiency (measured in Solve Frequency per Watt (SFPW)) of different solvers in dynamic trotting over different target computers, planning horizons ($N = 10$ and $N = 20$), and WBC formulations. The most efficient solver per target computer is marked in bold. The results the show that the efficiency of the solvers per target system scales inversely with the solve time. Therefore, HPIPM speed_abs has the highest efficiency on all systems for MPC and Eiquadprog in case of the WBC.

When comparing the efficiency of the three target computers, the Jetson Orin performs best: In case of MPC, it is more than twice as efficient as the LattePanda and around three times as efficient as the desktop PC. In case of WBC, it is roughly four times faster than the others.

V. DISCUSSION

The experimental results provide valuable insights into the performance of different solvers in dynamic quadrupedal walking regarding (1) sparsity, (2) variance over different tasks, and (3) HW efficiency.

TABLE IV
EFFICIENCY OF MPC AND WBC SOLVERS ON DIFFERENT HW

			Mean SFPW ($\mathrm{Hz\,W^{-1}}$)		
			Jetson Orin	Desktop	LattePanda
MPC	$N = 10$	DAQP	77.47	17.41	24.69
		HPIPM balance (sparse)	70.08	22.01	26.75
		HPIPM speed_abs (sparse)	**159.40**	**50.80**	**66.65**
		OSQP (sparse)	90.35	17.68	24.22
		qpOASES	93.94	18.70	25.33
	$N = 20$	DAQP	8.82	1.20	2.64
		HPIPM balance (sparse)	22.35	7.50	8.80
		HPIPM speed_abs (sparse)	**67.30**	**23.19**	**31.06**
		OSQP (sparse)	42.79	8.58	11.77
		qpOASES	11.36	2.46	2.20
WBC	Full TSID	Eiquadprog	**958.93**	**212.66**	**257.58**
		HPIPM	576.14	177.92	200.99
		Proxqp	795.34	190.80	216.11
		qpOASES	713.63	142.82	180.80
	Red. TSID	Eiquadprog	**1259.37**	**304.99**	**319.45**
		HPIPM	753.73	198.35	234.55
		Proxqp	954.99	231.86	265.41
		qpOASES	977.83	238.09	263.70

a) Sparsity: For larger QPs, as they occur in MPC, sparse formulations, when tackled by sparse solvers, generally seem to be advantageous over dense formulations. This effect becomes more significant with increasing prediction horizon. Here, condensing the QP and solving it has the opposite effect: the solution time increases significantly. This finding is in line with the theory of [21] and [12] that in MPC, if the input vector has almost the same size as the state vector, the fully sparse formulation is a good choice. For robots with fewer control inputs, however, condensing could improve performance. The results also indicate that for even smaller prediction horizons than $N = 10$ or smaller formulations, as in [13], ASM solvers such as qpOASES could be advantageous. The MPC problem analyzed here seems to be at the limit of the problem size where ASM is outperformed by other methods. It should also be considered that the HPIPM speed_abs solver, which outperforms the other solvers even at $N = 10$, does not provide accurate results for some applications. In the case of WBC, where significantly smaller problems are considered, the dense solvers and especially ASM perform well for the reduced formulation. The results for the full TSID formulation show that if the problem gets bigger, other methods such as IPM could indeed be considered as an option. In contrast to MPC, the choice of the QP solver is not of great relevance for WBC problems. Instead, efforts should be directed towards researching robust and stable QP formulations for WBC.

b) Variance over different tasks: The comparison of different motion tasks shows that in static cases such as standing, the ASM-based solvers, especially qpOASES, has a lower solve time than when trotting. This is in line with the general finding that ASM benefit from a stable problem structure, as the active-set can be warm-started [33], [34]. The same finding applies to OSQP. IPM solvers, here HPIPM, show a certain robustness against changing problem structures, as already stated by [10]. In contrast to all other solvers, HPIPM does not exhibit a second peak in the solve time histogram while the robot is trotting. The compromise could be that, for robot tasks where the problem is very dynamic, IPM provides constant performance, while for in static tasks ASM or ALM might be advantageous.

c) HW efficiency: The comparison of the different computer architectures shows that the solution times of the Jetson Orin and the Latte Panda are comparable, while the desktop delivers significantly faster solutions. However, the efficiency measurement shows that the faster solution times come at the expense of power consumption, resulting in a comparable efficiency for both x86 systems, with the LattePanda being more efficient in the range of $10\,\mathrm{Hz\,W^{-1}}$ depending on the prediction horizon and selected solver. On the other hand, the Jetson Orin is at least twice as efficient as the Desktop, and for the fastest solver and $N = 10$ even up to three times as efficient. This makes the Jetson Orin and ARM an ideal platform for applications with energy constraints, which is almost always the case in legged robotics.

VI. CONCLUSIONS

This work describes a benchmark for QP solvers in MPC and WBC in dynamic quadrupedal walking. The benchmark involves different QP formulations (sparse to dense), several solvers, robot tasks and HW architectures.

From this work's findings, three main conclusions follow: (1) For MPC, sparse solvers and especially solvers based on IPM (here HPIPM) perform best in dynamic quadrupedal walking and should be considered especially for long prediction horizons. These solvers also show certain robustness to changing problem structures, e.g., when changing contacts or between different tasks, and are therefore better suited for dynamic quadrupedal walking than other methods. (2) In WBC, any of the regarded open-source solvers performs well; the engineering effort should rather be put into the formulation of the WBC problem itself. (3) ARM architecture (here Jetson Orin) shows better efficiency than x86 when considering the Solve Frequency per Watt as a metric. Thus, they should be preferred in resource-constrained applications like autonomous quadrupedal walking.

Future work includes extending the benchmark to other HW architectures (e.g. CUDA), additional solvers to verify the results obtained, and possibly more complex systems (e.g. humanoids) to investigate the impact of system complexity on performance.

REFERENCES

[1] M. Hutter, C. Gehring, A. Lauber, F. Gunther, C. D. Bellicoso, V. Tsounis, P. Fankhauser, R. Diethelm, S. Bachmann, M. Bloesch, H. Kolvenbach, M. Bjelonic, L. Isler, and K. Meyer, "ANYmal - toward legged robots for harsh environments," *Advanced Robotics*, vol. 31, no. 17, pp. 918–931, Sept. 2017, publisher: Taylor

& Francis _eprint: https://doi.org/10.1080/01691864.2017.1378591. [Online]. Available: https://doi.org/10.1080/01691864.2017.1378591

[2] B. Dynamics, "Boston Dynamics Spot," 2024. [Online]. Available: https://bostondynamics.com/products/spot/

[3] G. Robotics, "Ghost Robotics VISION 60," 2024. [Online]. Available: https://www.ghostrobotics.io/vision-60

[4] S. Katayama, M. Murooka, and Y. Tazaki, "Model predictive control of legged and humanoid robots: models and algorithms," *Advanced Robotics*, vol. 37, no. 5, pp. 298–315, Mar. 2023, publisher: Taylor & Francis _eprint: https://doi.org/10.1080/01691864.2023.2168134. [Online]. Available: https://doi.org/10.1080/01691864.2023.2168134

[5] J. Carpentier and P.-B. Wieber, "Recent Progress in Legged Robots Locomotion Control," *Current Robotics Reports*, vol. 2, no. 3, pp. 231–238, Sept. 2021. [Online]. Available: https://doi.org/10.1007/s43154-021-00059-0

[6] U. Robotics, "Unitree Go2 Quadruped." [Online]. Available: https://www.unitree.com/go2

[7] H. J. Ferreau, C. Kirches, A. Potschka, H. G. Bock, and M. Diehl, "qpOASES: a parametric active-set algorithm for quadratic programming," *Mathematical Programming Computation*, vol. 6, no. 4, pp. 327–363, Dec. 2014. [Online]. Available: https://doi.org/10.1007/s12532-014-0071-1

[8] B. Stellato, G. Banjac, P. Goulart, A. Bemporad, and S. Boyd, "OSQP: An Operator Splitting Solver for Quadratic Programs," *Mathematical Programming Computation*, vol. 12, no. 4, pp. 637–672, Dec. 2020, arXiv:1711.08013 [math]. [Online]. Available: http://arxiv.org/abs/1711.08013

[9] S. Boyd, N. Parikh, E. Chu, B. Peleato, and J. Eckstein, "Distributed Optimization and Statistical Learning via the Alternating Direction Method of Multipliers," *Foundations and Trends® in Machine Learning*, vol. 3, no. 1, pp. 1–122, July 2011, publisher: Now Publishers, Inc. [Online]. Available: https://www.nowpublishers.com/article/Details/MAL-016

[10] G. Frison and M. Diehl, "HPIPM: a high-performance quadratic programming framework for model predictive control∗," *IFAC-PapersOnLine*, vol. 53, no. 2, pp. 6563–6569, Jan. 2020. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S2405896320303293

[11] J. L. Jerez, E. C. Kerrigan, and G. A. Constantinides, "A sparse and condensed QP formulation for predictive control of LTI systems," *Automatica*, vol. 48, no. 5, pp. 999–1002, May 2012. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0005109812001069

[12] D. Axehill, "Controlling the level of sparsity in MPC," *Systems & Control Letters*, vol. 76, pp. 1–7, Feb. 2015. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0167691114002680

[13] J. Di Carlo, P. M. Wensing, B. Katz, G. Bledt, and S. Kim, "Dynamic Locomotion in the MIT Cheetah 3 Through Convex Model-Predictive Control," in *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, Oct. 2018, pp. 1–9, iSSN: 2153-0866. [Online]. Available: https://ieeexplore.ieee.org/document/8594448

[14] D. Kim, J. Di Carlo, B. Katz, G. Bledt, and S. Kim, "Highly Dynamic Quadruped Locomotion via Whole-Body Impulse Control and Model Predictive Control," Sept. 2019, arXiv:1909.06586 [cs]. [Online]. Available: http://arxiv.org/abs/1909.06586

[15] D. Dimitrov, A. Sherikov, and P.-B. Wieber, "A sparse model predictive control formulation for walking motion generation," in *2011 IEEE/RSJ International Conference on Intelligent Robots and Systems*, Sept. 2011, pp. 2292–2299, iSSN: 2153-0866. [Online]. Available: https://ieeexplore.ieee.org/abstract/document/6095035

[16] S. Caron, A. Zaki, P. Otta, D. Arnström, J. Carpentier, and F. Yang, "qpbenchmark: Benchmark for quadratic programming solvers available in Python," Feb. 2024. [Online]. Available: https://github.com/qpsolvers/qpbenchmark

[17] C. C. Attila Kozma and M. Diehl, "Benchmarking large-scale distributed convex quadratic programming algorithms," *Optimization Methods and Software*, vol. 30, no. 1, pp. 191–214, 2015. [Online]. Available: https://doi.org/10.1080/10556788.2014.911298

[18] A. Spiridonov, F. Buehler, M. Berclaz, V. Schelbert, J. Geurts, E. Krasnova, E. Steinke, J. Toma, J. Wuethrich, R. Polat, W. Zimmermann, P. Arm, N. Rudin, H. Kolvenbach, and M. Hutter, "SpaceHopper: A Small-Scale Legged Robot for Exploring Low-Gravity Celestial Bodies," Mar. 2024, arXiv:2403.02831 [cs]. [Online]. Available: http://arxiv.org/abs/2403.02831

[19] P. Arm, R. Zenkl, P. Barton, L. Beglinger, A. Dietsche, L. Ferrazzini, E. Hampp, J. Hinder, C. Huber, D. Schaufelberger, F. Schmitt, B. Sun, B. Stolz, H. Kolvenbach, and M. Hutter, "SpaceBok: A Dynamic Legged Robot for Space Exploration," in *2019 International Conference on Robotics and Automation (ICRA)*. Montreal, QC, Canada: IEEE, May 2019, pp. 6288–6294. [Online]. Available: https://ieeexplore.ieee.org/document/8794136/

[20] Y. Ding, A. Pandala, and H.-W. Park, "Real-time Model Predictive Control for Versatile Dynamic Motions in Quadrupedal Robots," in *2019 International Conference on Robotics and Automation (ICRA)*, May 2019, pp. 8484–8490, iSSN: 2577-087X. [Online]. Available: https://ieeexplore.ieee.org/abstract/document/8793669

[21] G. Frison, D. Kouzoupis, J. B. Jørgensen, and M. Diehl, "An efficient implementation of partial condensing for Nonlinear Model Predictive Control," in *2016 IEEE 55th Conference on Decision and Control (CDC)*, Dec. 2016, pp. 4457–4462. [Online]. Available: https://ieeexplore.ieee.org/abstract/document/7798946

[22] A. Del Prete, N. Mansard, O. E. Ramos, O. Stasse, and F. Nori, "Implementing Torque Control with High-Ratio Gear Boxes and Without Joint-Torque Sensors," *International Journal of Humanoid Robotics*, vol. 13, no. 01, p. 1550044, Mar. 2016. [Online]. Available: https://www.worldscientific.com/doi/abs/10.1142/S0219843615500449

[23] S. Hemmert, "Green HPC: From Nice to Necessity," *Computing in Science & Engineering*, vol. 12, no. 6, pp. 8–10, Nov. 2010. [Online]. Available: http://ieeexplore.ieee.org/document/5624674/

[24] R. Tedrake and Drake-Development-Team, "Drake: Model-based design and verification for robotics," 2019. [Online]. Available: https://drake.mit.edu

[25] R. Verschueren, G. Frison, D. Kouzoupis, J. Frey, N. v. Duijkeren, A. Zanelli, B. Novoselnik, T. Albin, R. Quirynen, and M. Diehl, "acados—a modular open-source framework for fast embedded optimal control," *Mathematical Programming Computation*, vol. 14, no. 1, pp. 147–183, Mar. 2022. [Online]. Available: https://doi.org/10.1007/s12532-021-00208-8

[26] D. Mronga, "ARC-OPT: Adaptive Robot Control using Optimization," Sept. 2024. [Online]. Available: https://github.com/ARC-OPT/wbc/

[27] D. Mronga, S. Kumar, and F. Kirchner, "Whole-Body Control of Series-Parallel Hybrid Robots," in *2022 International Conference on Robotics and Automation (ICRA)*, May 2022, pp. 228–234. [Online]. Available: https://ieeexplore.ieee.org/document/9811616

[28] D. Arnstrom, A. Bemporad, and D. Axehill, "A Dual Active-Set Solver for Embedded Quadratic Programming Using Recursive LDL$ˆ{T}$ Updates," *IEEE Transactions on Automatic Control*, vol. 67, no. 8, pp. 4362–4369, Aug. 2022. [Online]. Available: https://ieeexplore.ieee.org/document/9779534/

[29] G. Buondonno, "eiquadprog," Toulouse, France, Oct. 2019, original-date: 2019-10-08T19:39:57Z. [Online]. Available: https://github.com/stack-of-tasks/eiquadprog

[30] A. Bambade, S. El-Kazdadi, A. Taylor, and J. Carpentier, "PROX-QP: Yet another Quadratic Programming Solver for Robotics and beyond," in *Robotics: Science and Systems XVIII*. Robotics: Science and Systems Foundation, June 2022. [Online]. Available: http://www.roboticsproceedings.org/rss18/p040.pdf

[31] D. Goldfarb and A. Idnani, "A numerically stable dual method for solving strictly convex quadratic programs," *Mathematical Programming*, vol. 27, no. 1, pp. 1–33, Sept. 1983. [Online]. Available: https://doi.org/10.1007/BF02591962

[32] G. Frison, H. H. B. Sørensen, B. Dammann, and J. B. Jørgensen, "High-performance small-scale solvers for linear Model Predictive Control," in *2014 European Control Conference (ECC)*, June 2014, pp. 128–133. [Online]. Available: https://ieeexplore.ieee.org/abstract/document/6862490

[33] S. Kuindersma, F. Permenter, and R. Tedrake, "An Efficiently Solvable Quadratic Program for Stabilizing Dynamic Locomotion," in *2014 IEEE International Conference on Robotics and Automation (ICRA)*, May 2014, pp. 2589–2594, arXiv:1311.1839 [cs]. [Online]. Available: http://arxiv.org/abs/1311.1839

[34] R. Bartlett, A. Wachter, and L. Biegler, "Active set vs. interior point strategies for model predictive control," in *Proceedings of the 2000 American Control Conference. ACC (IEEE Cat. No.00CH36334)*, vol. 6, June 2000, pp. 4229–4233 vol.6, iSSN: 0743-1619. [Online]. Available: https://ieeexplore.ieee.org/abstract/document/877018