Whole-Body Nonlinear Model Predictive Control Through Contacts for Quadrupeds

Michael Neunert, Markus Stäuble, Markus Giftthaler, Carmine D. Bellicoso, Jan Carius, Christian Gehring, Marco Hutter, and Jonas Buchli

***Abstract*—In this letter, we present a whole-body nonlinear model predictive control approach for rigid body systems subject to contacts. We use a full-dynamic system model which also includes explicit contact dynamics. Therefore, contact locations, sequences, and timings are not prespecified but optimized by the solver. Yet, using numerical and software engineering allows for running the nonlinear Optimal Control solver at rates up to 190 Hz on a quadruped for a time horizon of half a second. This outperforms the state-of-the-art by at least one order of magnitude. Hardware experiments in the form of periodic and nonperiodic tasks are applied to two quadrupeds with different actuation systems. The obtained results underline the performance, transferability, and robustness of the approach.**

***Index Terms*—Optimization and optimal control, legged robots, motion and path planning, force control.**

I. Introduction

In THIS letter, we present a whole-body Nonlinear Model Predictive Control (NMPC) approach for Rigid Body Dynamics (RBD) systems subject to contacts. By using an explicit, Auto-Differentiable contact model as part of the dynamic system, the approach is able to reason about contacts and optimize through them efficiently. Contact timings, sequences or locations are not pre-specified, but an outcome of the optimization. Thanks to a highly-efficient, unconstrained nonlinear optimal control solver, we are able to successfully apply the method to two different quadruped platforms. We verify the performance and versatility of our approach by testing a multitude of tasks including periodic gait patterns as well as highly dynamic motions, such as squat jumps.

A. Related Work

Motion planning and control for legged, and especially quadruped locomotion is often tackled with multi-stage planning and control frameworks [1]–[4]. Such frameworks often consist of one or multiple planner stages that use a simplified model and a reactive tracking controller. This results in the dilemma that the planner does not always produce feasible, i.e., dynamically consistent, plans or needs to plan conservatively. The tracking controller on the other hand does not have enough control authority to e.g., modify foothold positions or contact timings, but blindly tries to track the planners reference. In this field, centroidal dynamics approaches [5]–[9] become increasingly popular as they capture the core dynamics of the problem. However, many of these approaches plan or optimize contact forces that are not guaranteed to be realizable.

In recent years, there has been an increasing number of whole-body optimization and optimal control based approaches [4], [10]–[13]. While these approaches are very complete, their runtimes are still a few orders of magnitudes away from running in receding horizon or MPC fashion. There are also some whole-body NMPC approaches verified in simulation [14]. However, without hardware validation, it remains an open question if and how well these approaches can be applied to real robots. While whole-body, contact invariant NMPC has been demonstrated on hardware before [15], the presented motions were rather slow or even quasi static, underlined by the fact that the authors do not apply the torque output to the robot but instead only use the position and velocity trajectories as references for the joint controllers. Additionally, contact switching was not dynamic and artificially enforced. Also stability during execution was explicitly encoded in the task.

B. Contributions

In this work, we demonstrate whole-body, contact invariant nonlinear MPC for highly dynamic motions that require explicit reasoning about the full dynamics of the system and the contacts. Furthermore, to the best of our knowledge, we demonstrate that such a framework can be applied to both single motions as well as periodic gaits on hardware for the first time. We show that the approach transfers between platforms, and apply the same framework to the two quadrupeds HyQ and ANYmal (Fig. 1). We also demonstrate the robustness and replanning capabilities of the approach by adding significant disturbances during execution. We summarize our solver framework, which uses Auto-Differentiation and code generation to achieve high computational performance exceeding the current state of the art in robotics NMPC applications by at least one order of magnitude. In contrast to many previous approaches, our solver is also available as open-source software [16]. Furthermore, we

Manuscript received September 10, 2017; accepted December 31, 2017. Date of publication January 31, 2018; date of current version February 22, 2018. This letter was recommended for publication by Associate Editor B. Ugurlu and Editor N. Tsagarakis upon evaluation of the reviewers' comments. This work was supported in part by the Swiss National Science Foundation through the NCCR Digital Fabrication and in part by the NCCR Robotics. The work of J. Buchli was supported by a Professorship Award. (*Corresponding author: Michael Neunert.)*

M. Neunert, M. Stäuble, M. Giftthaler, and J. Buchli are with the Agile & Dexterous Robotics Laboratory, ETH Zürich, Zürich 8092, Switzerland (e-mail: neunertm@gmail.com; markusta@ethz.ch; mgiftthaler@ethz.ch; buchlij@ethz.ch).

C. D. Bellicoso, J. Carius, C. Gehring, and M. Hutter are with the Robotic Systems Laboratory, ETH Zürich, Zürich 8092, Switzerland (e-mail: bcarmine@ethz.ch; jcarius@ethz.ch; gehrinch@ethz.ch; mahutter@ethz.ch).

Digital Object Identifier 10.1109/LRA.2018.2800124

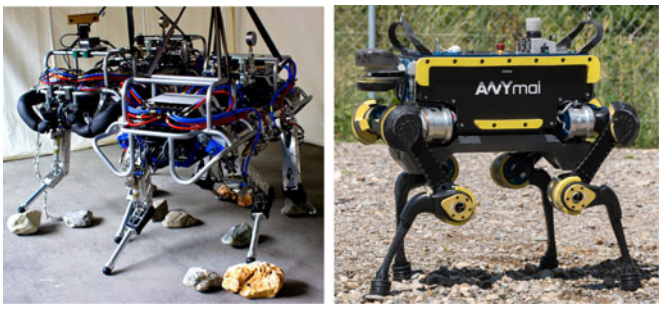

Fig. 1. The quadrupeds HyQ-blue (left front) and ANYmal (right), which served as test platforms for our MPC experiments.

also publish the cost function weights used during experiments to ensure reproducibility.

In previous work [17], we demonstrated the versatility of optimizing whole-body motions through contacts. However, trajectories were only optimized once before execution. Also, especially interesting tasks such as periodic gaits could not be transferred to hardware due to model mismatches and lack of robustness of the plans. In this work, we demonstrate that an NMPC approach which continuously re-optimizes the state and control trajectories at high frequency, results in robust performance and copes with model mismatches. Running NMPC on real hardware poses severe restrictions in terms of computation time and software integration. In this work, we describe how we overcome these issues and improve our solver to achieve a speedup of several orders of magnitude.

C. Structure of This Letter

We organize the letter as follows. In Section II, we define how we formulate the NMPC problem for Rigid Body Dynamics Systems. In Section III we describe our approach of solving the problem. Afterwards, in Sections IV and V, we describe the implementation from a software point of view as well as the integration on hardware. In Section VI, we then present our experiments and discuss the results and findings. Finally, in Section VII we summarize the letter and provide an outlook to future work.

II. NMPC for Rigid Body Systems

The NMPC approach used in this work recurrently solves finite-horizon Optimal Control Problems with cost functions of the form

$$J(\boldsymbol{x}(t), \boldsymbol{u}(t)) = h\left(\boldsymbol{x}(t_f)\right) + \int_{t=0}^{t_f} L\left(\boldsymbol{x}(t), \boldsymbol{u}(t), t\right) dt \quad (1)$$

and nonlinear system dynamics

$$\dot{\boldsymbol{x}}(t) = f(\boldsymbol{x}(t), \boldsymbol{u}(t), t), \quad \boldsymbol{x}(0) = \boldsymbol{x}_0 \quad (2)$$

with state and control trajectories $\boldsymbol{x}(t)$ and $\boldsymbol{u}(t)$.

A. System Modelling

Analogously to [17], we consider general Rigid Body Dynamics of the form

$$\boldsymbol{M}(\boldsymbol{q})\ddot{\boldsymbol{q}} + \boldsymbol{C}(\boldsymbol{q}, \dot{\boldsymbol{q}}) + \boldsymbol{G}(\boldsymbol{q}) = \boldsymbol{J}_c^\top \boldsymbol{\lambda}(\boldsymbol{q}, \dot{\boldsymbol{q}}) + \boldsymbol{S}^\top \boldsymbol{\tau} \quad (3)$$

where $\boldsymbol{M}_{18\times18}$ is the inertia matrix, $\boldsymbol{C}_{18\times1}$ captures Coriolis and centripetal forces and $\boldsymbol{G}_{18\times1}$ represents the gravity terms. We further assume the system is actuated via joint forces/torques $\boldsymbol{\tau}_{12\times1}$ where the selection matrix $\boldsymbol{S}_{18\times12}$ maps these forces to the actuated degrees of freedom. Additionally, external forces $\boldsymbol{\lambda}_{12\times1}$ act on the system via the contact Jacobian $(\boldsymbol{J}_c)_{18\times12}$. The joint positions/angles and their velocities are denoted by $\boldsymbol{q}$ and $\dot{\boldsymbol{q}}$ respectively. In the case of a floating base robot, these quantities also contain the base pose (i.e., its orientation and position) as well as the base twist (i.e., its linear and angular velocity). These quantities can also be thought of as an unactuated 6 DoF joint between an inertial and the base frame. We use an Euler-Angle parametrization of the 3D orientation.

We define the state of our system as follows

$$\boldsymbol{x} = [{}_{\mathcal{W}}\boldsymbol{q}^\top \; {}_{\mathcal{L}}\dot{\boldsymbol{q}}^\top]^\top = [{}_{\mathcal{W}}\boldsymbol{q}_B^\top \; {}_{\mathcal{W}}\boldsymbol{x}_B^\top \; \boldsymbol{q}_J^\top \; {}_{\mathcal{L}}\boldsymbol{\omega}_B^\top \; {}_{\mathcal{L}}\dot{\boldsymbol{x}}_B^\top \; \dot{\boldsymbol{q}}_J^\top]^\top \quad (4)$$

where the pose is expressed in an inertial "world" frame $\mathcal{W}$ and the twist is expressed in a local body frame $\mathcal{B}$. Due to the different reference frames, the twist is not a pure time derivative of the pose, but requires an additional coordinate transform $\boldsymbol{T}_{\mathcal{WL}}$ which is composed of a pure rotation matrix for linear velocities as well as a slightly more complex mapping matrix for angular velocities. This leads to the overall system dynamics

$$\begin{aligned}\dot{\boldsymbol{x}}(t) &= \begin{bmatrix} {}_{\mathcal{W}}\dot{\boldsymbol{q}}^\top & {}_{\mathcal{L}}\ddot{\boldsymbol{q}}^\top \end{bmatrix}^\top \\ &= \begin{bmatrix} \boldsymbol{T}_{\mathcal{WL}}\, {}_{\mathcal{L}}\dot{\boldsymbol{q}} \\ \boldsymbol{M}^{-1}(\boldsymbol{q})(\boldsymbol{S}^\top\boldsymbol{\tau} + \boldsymbol{J}_c\boldsymbol{\lambda}(\boldsymbol{q}, \dot{\boldsymbol{q}}) - \boldsymbol{C}(\boldsymbol{q}, \dot{\boldsymbol{q}}) - \boldsymbol{G}(\boldsymbol{q})) \end{bmatrix}\end{aligned} \quad (5)$$

B. Contact Model

In order to avoid pre-specifying contact sequences, locations or timings, we need to enable the NMPC solver to reason about contacts. Some approaches resort to adding complementarity constraints to enforce contacts (e.g., [18]). However, these constraints do not satisfy the Linear Independence Constraint Qualification (LICQ) [19]. Almost all off-the-shelf Nonlinear Optimal Control or Nonlinear Programming solvers assume LICQ and, therefore, cannot handle these problems [20]. In contrast, we add the contact physics to our dynamic model using an explicit contact model. As a result, the contact forces become an explicit function of the robot's state: $\boldsymbol{\lambda}(\boldsymbol{q}, \dot{\boldsymbol{q}}) = \boldsymbol{g}(\boldsymbol{x}(t))$.

Our contact model consists of a combination of linear springs and dampers perpendicular and parallel to the contact surface. For each end-effector we compute the contact model in the specialized contact frame C as follows:

$$\begin{aligned}{}_C\boldsymbol{\lambda}(\boldsymbol{q}, \dot{\boldsymbol{q}}) = \; & -k \exp(\alpha_k \, {}_C\boldsymbol{p}_z(\boldsymbol{q})) \\ & -d \, \mathrm{sig}(\alpha_d \, {}_C\boldsymbol{p}_z(\boldsymbol{q})) \, {}_C\dot{\boldsymbol{p}}(\boldsymbol{q}, \dot{\boldsymbol{q}})\end{aligned} \quad (6)$$

with d and k being damper and spring parameters. In order to achieve smooth derivatives, we multiply the damper-term with a sigmoid function of the normal component $\boldsymbol{p}_z(\boldsymbol{q})$ of the contact surface penetration $\boldsymbol{p}(\boldsymbol{q})$. Both the exponential and the sigmoid function serve as smoothing elements. Their 'sharpness' is controlled by α_k and α_d. Finally, the contact forces are transformed into the robot body frame by

$${}_B\boldsymbol{\lambda}(\boldsymbol{q}, \dot{\boldsymbol{q}}) = \boldsymbol{R}_{W\,B}(\boldsymbol{q}) \, {}_C\boldsymbol{\lambda}(\boldsymbol{q}, \dot{\boldsymbol{q}}) \quad (7)$$

and subsequently passed to the forward dynamics where they act on the corresponding link. Our particular choice of the contact model smoothing supports the gradient-based solver by ensuring that contact forces never completely vanish. Therefore, the solver can reason about contacts even before they are established. While this is slightly nonphysical, we will later see that this does not hinder good performance on hardware. We emphasize that there exists physically more accurate explicit contact models [21] and implicit contact models as popular in physics engines. The latter however rely on optimization based solvers which cannot be differentiated well. In contrast, our simplified model captures the governing effects accurately enough and allows for computing derivatives efficiently by using Auto-Diff which is key to solving the NMPC problem fast enough [22].

III. NMPC Approach

Informally speaking, NMPC is achieved through solving the Nonlinear Optimal Control (NLOC) problem at sufficiently high rates. Popular approaches to nonlinear optimal control are Single Shooting, Multiple Shooting or Direct Collocation [23], which discretize the continuous time optimal control problem and transcribe it into a Nonlinear Program (NLP). These NLPs are often solved using general off-the-shelf NLP solvers as presented in [24], [25]. However, such an approach does not fully exploit the sparsity structure inherent to optimal control problems and often results in poor algorithm runtimes, which are not fast enough for MPC applications.

To overcome this issue, we formulate our problem as an unconstrained optimal control problem, and resort to an optimized, custom solver, that implements a family of iterative Gauss-Newton NLOC algorithms [26]. The solver employs a first-order method that locally approximates the NLOC problem as a Linear Quadratic Optimal Control (LQOC) problem using a Gauss-Newton Hessian approximation. The LQOC is solved by a Riccati-based solver which has linear complexity in the time horizon. This makes the approach efficient for larger time horizons. Our solver can be considered a generalization of the well-known iLQR [27] and SLQ [28] algorithms and covers both Single and Multiple Shooting. It designs time-varying state-feedback controllers of the form

$$\boldsymbol{u}_n(\boldsymbol{x}) = \boldsymbol{u}_n^{ff} + \boldsymbol{K}_n(\boldsymbol{x}_n - \boldsymbol{x}_n^{\text{ref}}) \tag{8}$$

where $\boldsymbol{u}_n^{ff}$ is the feedforward control action and $\boldsymbol{K}_n$ a linear feedback controller regulating deviations of the state $\boldsymbol{x}_n$ from the reference trajectory $\boldsymbol{x}_n^{\text{ref}}$. For most experiments in this letter, we use the iLQR algorithm. We furthermore compare it to the more efficient Gauss-Newton Multiple Shooting (GNMS) approach which can act as a direct replacement. Both algorithms use the same approach of formulating and solving a local LQOC problem, however their MPC formulations varies.

A summary of the iLQR-NMPC algorithm, is given in Algorithm 1. It shows two forward integration steps during the algorithm. One directly after retrieving the state measurement to get the nominal trajectory, and a second one during the line search after updating the controller. For our application, the latter is important to obtain a new reference trajectory for the feedback controller to track. In contrast, the GNMS-NMPC algorithm, which is summarized in Algorithm 2, designs a state reference trajectory simultaneously with the new control policy. Furthermore, it allows to separate the algorithm into a *feedback* and a *preparation* phase [29], which helps to minimize the latency between state-measurement and control policy update. For a detailed overview about the GNMS algorithm, the reader is referred to [26]. An open-source reference implementation is available in [16].

Algorithm 1: Discrete-time iLQR-MPC Algorithm.

Given
- cost function (1) and system dynamics (2).
- receding MPC time horizon N.
- stable initial control policy $\boldsymbol{u}_n(\boldsymbol{x})$ of form (8)

Repeat Online:
- get state measurement $\boldsymbol{x}_{\text{meas}}$.
- forward integrate system dynamics (2) with $\boldsymbol{x}_0 = \boldsymbol{x}_{\text{meas}}$ to obtain state trajectories $\boldsymbol{X} = \{\boldsymbol{x}_0, \boldsymbol{x}_1, \ldots, \boldsymbol{x}_N\}$, control trajectories $\boldsymbol{U} = \{\boldsymbol{u}_0, \boldsymbol{u}_1, \ldots, \boldsymbol{u}_{N-1}\}$ and corresponding sensitivities $\boldsymbol{A}_n,\ \boldsymbol{B}_n$.
- quadratize cost function (1) around $\boldsymbol{X}$ and $\boldsymbol{U}$
- solve LQOC problem using a Riccati backward sweep
- retrieve control policy $\boldsymbol{u}_n^+(\boldsymbol{x})$ of form (8)
- **line search** over the control increment $(\boldsymbol{u}_n(\boldsymbol{x})^+ - \boldsymbol{u}_n(\boldsymbol{x}))$ and update $\boldsymbol{X}^+$ by means of a forward simulation of the nonlinear dynamics (2) with $\boldsymbol{x}_0 = \boldsymbol{x}_{\text{meas}}$
- send policy $\boldsymbol{u}_n^+(\boldsymbol{x})$ and $\boldsymbol{X}^+$ to the robot tracking controller
- update $\boldsymbol{u}_n(\boldsymbol{x}) \leftarrow \boldsymbol{u}_n^+(\boldsymbol{x})$

Algorithm 2: Discrete-time GNMS-NMPC Algorithm.

Given
- cost function (1) and system dynamics (2).
- receding MPC time horizon N.
- initial state and control trajectories $\boldsymbol{X} = \{\boldsymbol{x}_0, \boldsymbol{x}_1, \ldots, \boldsymbol{x}_N\}, \boldsymbol{U} = \{\boldsymbol{u}_0, \boldsymbol{u}_1, \ldots, \boldsymbol{u}_{N-1}\}$

Repeat Online:

Feedback phase
- get state measurement $\boldsymbol{x}_{\text{meas}}$.
- forward integrate system dynamics (2) with $\boldsymbol{x}_0 = \boldsymbol{x}_{\text{meas}}$ on the first multiple-shooting interval, obtain sensitivities $\boldsymbol{A}_0,\ \boldsymbol{B}_0$.
- quadratize cost function (1) around $\boldsymbol{X}$ and $\boldsymbol{U}$ for first control stage
- solve LQOC problem using a Riccati backward sweep
- retrieve updated control policy $\boldsymbol{u}_n^+(\boldsymbol{x})$ and updated trajectories $\boldsymbol{U}^+, \boldsymbol{X}^+$.
- send policy $\boldsymbol{u}_n^+(\boldsymbol{x})$ and $\boldsymbol{X}^+$ to the robot tracking controller

Preparation phase
- update: $\boldsymbol{u}_n(\boldsymbol{x}) \leftarrow \boldsymbol{u}_n^+(\boldsymbol{x})$, $\boldsymbol{X} \leftarrow \boldsymbol{X}^+, \boldsymbol{U} \leftarrow \boldsymbol{U}^+$
- forward integrate system dynamics (2) for the multiple-shooting intervals 1 to N, obtain sensitivities $\boldsymbol{A}_1, \ldots \boldsymbol{A}_{N-1}, \boldsymbol{B}_1, \ldots, \boldsymbol{B}_{N-1}$.
- quadratize cost function (1) around $\boldsymbol{X}, \boldsymbol{U}$ for multiple-shooting intervals 1 to N.

IV. Software Implementation

Running NMPC for a high dimensional system in real-time remains a challenge despite the powerful consumer PCs available today. While the development of processors with faster clock speed has stalled in recent years, processing power instead foremost grows due to higher computation core counts as well as vectorization. However, both parallel execution and vectorization cannot be leveraged automatically by standard compilers. Also, many computational routines such as integrating a differential equation over time, are naturally sequential operations that cannot be parallelized easily. In this subsection we describe how we optimize the NMPC solver from a numerical point of view and leverage the processor architecture to reduce the computational burden.

A. Modelling Framework

Our NMPC controller relies heavily on evaluating Rigid Body Dynamics and Kinematics. Therefore, we use RobCoGen [30], an efficient code generation framework for modelling Rigid Body Dynamics. In [22] we augmented RobCoGen to be compatible with Auto-Differentiation as implemented in our Control Toolbox (CT) [16]. Furthermore, we use CT's contact model and kinematics that wrap around RobCoGen to provide contact force mappings. The resulting framework is lean compared to sophisticated physics engines and produces fast to evaluate, hard realtime capable code.

B. Integration and Sensitivity Computation

Our system dynamics include a contact model that needs to be chosen stiff enough to approximate the real physics of contact well. Naturally, this leads to a numerical stiffness which requires us to take small integration time-steps. Therefore, instead of using standard explicit integrators such as Euler or Runge-Kutta schemes, we use symplectic (or semi-implicit) integrators, which are also popular in physics engines [31]. A symplectic integrator alternates between integrating positions and velocities, i.e., the updated positions are used to compute the velocity update and vice versa. Therefore, the computational complexity is similar to an explicit scheme, but the numerical stability is increased. In contrast to implicit integrators, we do not have to compute Jacobians explicitly and do not have to solve an internal optimization problem. The symplectic integrator allows us to increase the integration step size by a factor of four compared to explicit schemes [17].

Our LQOC solver requires us to compute sensitivities along the trajectory, i.e., partial derivatives of the integrated state with respect to the start state and control action of the step. These sensitivities can be understood as matrices $\boldsymbol{A}_n$ and $\boldsymbol{B}_n$ in a local linear approximation of the form $\boldsymbol{x}_{n+1} = \boldsymbol{A}_n \boldsymbol{x}_n + \boldsymbol{B}_n \boldsymbol{u}_n + \boldsymbol{c}_n$. While many existing approaches compute the sensitivities numerically [15], it is slow and can lead to severe numerical problems hindering convergence [32]. Therefore, we compute them exactly by integrating a corresponding sensitivity ODE. A description of the integration scheme for sensitivities of symplectic integrators can be found in [33]. The linearization of our dynamic system is performed with Auto-Diff and code-generation described in detail in [22], which provides the same accuracy as analytic derivatives but outperforms them in terms of computational speed.

C. Multithreading and Vectorization

Another important factor for obtaining best performance is multi-threading. Some parts of our algorithm are inherently sequential, for example solving the LQOC problem backwards in time, or the forward simulation of the system when employing iLQR. However, using a multiple-shooting approach allows us to parallelize the forward simulation over the individual multiple-shooting intervals. In this case, computation time decreases linearly with the number of cores. In contrast, iLQR requires to compute a single, continuous forward simulation and thus does not benefit from a multi-core processor in this step. The cost and sensitivity computation, which can be distributed among all available cores, is parallelizable for all our algorithm variants.

Another optimization possibility is to use the processor's vectorization capabilities, which are Single Instruction Multiple Data (SIMD) implementations, especially useful for mathematical operations on vectors and matrices. In a previous implementation [17], we had already used SSE [34] instructions. In this implementation, we switched to AVX [34] instructions. Since our code mostly consists of matrix and vector manipulations and register sizes of AVX are doubled over SSE, we obtained an additional speedup of almost a factor of two. This number is also supported by the release notes of the Eigen library [35] used for all computations. With the release of AVX512 doubling register sizes yet again, we expect another speedup of about factor of two.

V. Hardware Integration

A. Platform Descriptions

For hardware experiments we use two different quadruped robots, that strongly vary in size, weight and actuation principles. Therefore, the experiments underline the generality of the approach and show that no platform specific modifications are required. HyQ is an 80 kg, hydraulically actuated quadruped, while ANYmal weighs around 34 kg and uses electric, series-elastic actuators. Both robots are briefly described in the following.

1) HyQ: HyQ [36] is a fully torque controlled, hydraulic quadruped built by the Italian Institute of Technology. It features three joints per leg, namely Hip Abduction Adduction (HAA), Hip Flexion Extension (HFE) and Knee Flexion Extension (KFE). All joints are equipped with absolute and relative encoders, the joint torques are measured by load cells.

2) ANYmal: While being more compact, ANYmal [37] features the same joint configuration as HyQ. Furthermore, it is fully torque controlled via series-elastic actuators. Joint encoders before and after the elastic element measure its deflection which is used to compute the internal joint torque.

B. Tracking Controller

The control and tracking framework mostly corresponds to the one shown in Fig. 2 also used in [17]. When running our NMPC framework, we directly apply the optimized torques as feedforward signal to the actuators. However, in our framework, there is an average delay of about 10–15 ms between state measurement and execution of the corresponding, optimized policy on the robot. This delay prevents from robustly controlling

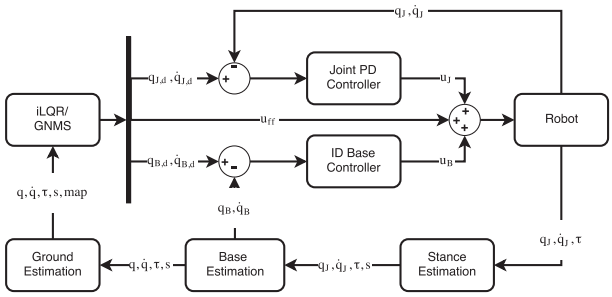

Fig. 2. Structure of the estimation and control approach for hardware execution of the NMPC controller. Estimators estimate ground height and base state information. The optimized control input obtained from the NMPC solver is then augmented with the output of two tracking controllers.

positions and velocities of the swing legs. Therefore, we add a PD control loop around the optimized joint trajectories that improves position and velocity tracking. Lastly, we also add a virtual model controller [38] using inverse dynamics (without gravity compensation) which only contributes a few percent of the overall control signal. Therefore, the presented tasks also work without the base controller but show slightly improved performance with the controller enabled.

C. State Estimation

Both quadrupeds run a state estimator that fuses measurements from an Inertial Measurement Unit (IMU) and the leg encoders to estimate the base pose and twist of the robot. The state estimator is described in [39]. Joint position measurements are directly obtained from both robots and are then numerically differentiated to obtain joint velocities. Since we use an explicit contact model, an estimation of the ground is required. Here we check the stance of each foot and fit a plane through all stance legs. This ground estimator could possibly be replaced in the future by a mapping approach that delivers differentiable height information.

D. Computing Setup

In our setup, we use a dedicated computer that runs the NMPC control loop. The NMPC-node receives the current state of the robot via the Robot Operating System (ROS) from the midlevel control computer that executes the tracking controller described in Subsection V-B. Due to different software and hardware architectures, this tracking controller runs at 250 Hz on HyQ and at 400 Hz on ANYmal. The midlevel controller then sends desired torque, position and velocity setpoints to a lowlevel torque controller. In return, it receives current state measurements and computes the base and ground state estimate. The lowlevel controller runs a torque tracking and a position PD controller at 1 kHz on HyQ and 2.5 kHz on ANYmal. This controller is implemented on embedded hard real-time systems on both robots.

VI. Results

The performance of our algorithms is assessed on both quadrupeds. We test a periodic trotting gait on both robots and disturb them during the tests. Furthermore, we execute additional dynamic motions on ANYmal. For all experiments, the cost functions weights are visualized in Fig. 3 and provided as supplementary material.

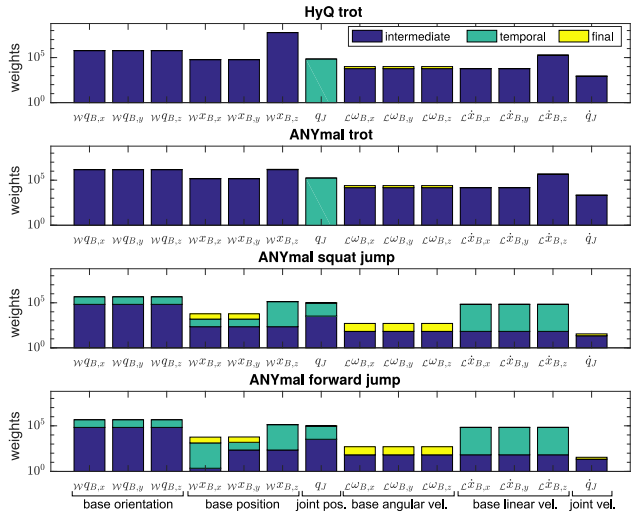

Fig. 3. Cost function weights for the different hardware experiments plotted on a logarithmic scale. All weighting matrices are diagonal and thus the weights illustrated correspond to the diagonal entries. Weights on joint positions q_J and velocities $\dot{q}_J$ are the same for all legs. The temporal costs in the trotting experiment affect swing leg pairs and are activated periodically. For the squat and forward jump temporal costs affect all legs equally and are activated once per jump.

In all experiments, we employ a time horizon of 500 ms, a control discretization of 4 ms and an integration rate of 1 ms for the aforementioned symplectic Euler scheme. In this work we employ the iLQR and GNMS solvers from [26], and use the solution from the previous MPC iteration as a warm start. We run a so called "real-time iteration scheme" [23], where we apply the optimized trajectory after a single iteration. Note that running only a single solver iteration before updating the state measurement results in better overall performance than running multiple iterations and letting the solver converge. All results are obtained with the same settings, solver and model. The different behaviors are thus simply a result of the choice of cost function and weights, offering a great versatility.

A. Hardware Experiments HyQ

1) Trotting: As a first test, we investigate a periodic gait pattern, encouraged over periodically activated costs penalizing joint angle deviations from a desired swing leg apex configuration. In previous work [17], we have demonstrated that our approach can also discover a trotting gait without swing leg costs. However, by adding such costs, we can influence the gait frequency and encourage trotting while staying in place. Since we only penalize the apex height of the swing leg, exact contact timings are not specified but can be adjusted by the algorithm. Additionally, since the swing phase is not a constraint, it can be violated by the algorithm if helpful for the overall performance. This can be observed during execution. If the robot base is strongly pushed to one side, the feet on that side are not lifted but remain on the ground to first stabilize the base. This means the algorithm naturally modifies the gait pattern and contact timings to optimize performance. Another observation is that, due to a feedforward dominant controller that plans ahead of time instead of simply reacting aggressively to disturbances, we obtain a compliant controller. HyQ can be perturbed significantly both on the base and the legs without reacting stiffly.

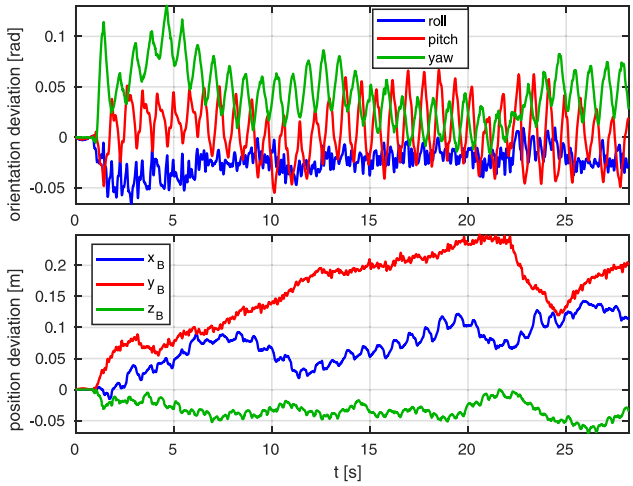

Fig. 4. Plots of the deviation of the orientation and position offset during trotting experiments on HyQ. In the cost function, we penalize orientation stronger than position which shows in the plots. Compared to ANYmal the magnitude of the deviations is slightly larger. However, this is in part also due to the different sizes of both robots.

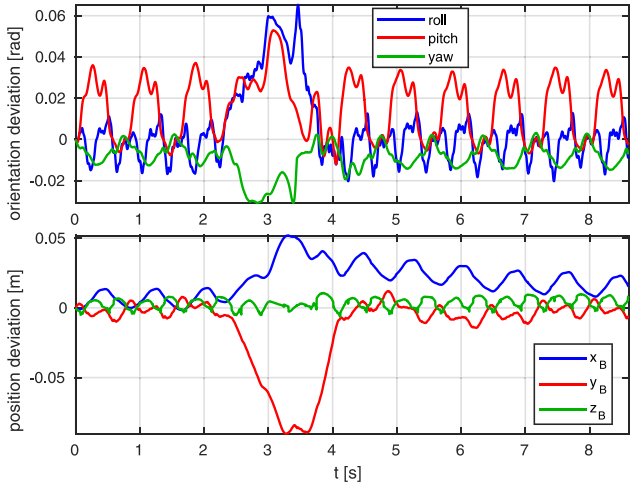

Fig. 5. Base orientation (top) and position (bottom) of ANYmal during a trot. At t = 2.8 s we placed a wooden stick below one foot which acted as heavy disturbance on the system. The controller is able to return to a periodic motion after the disturbance is removed.

Even placing planks under single feet does not deteriorate performance. Fig. 4 shows the results of a trotting experiment on HyQ. The plot shows that the base orientation and position deviations from the set point are regulated and remain small. Also, we add a strong cost penalty on the base orientation to improve stability. The resulting overall controller is stable and can robustly handle aforementioned disturbances.

B. Hardware Experiments ANYmal

1) Trotting: We repeat the same trotting experiments with adjusted weights (due to different mass/inertias) on ANYmal. Also here we observe that the controller is robust to disturbances. By adjusting the desired position and orientation of the base with a joystick, we navigate the robot around. Fig. 5 show measurements of the robot base during the trotting experiment. These plots illustrate how the MPC controller deals

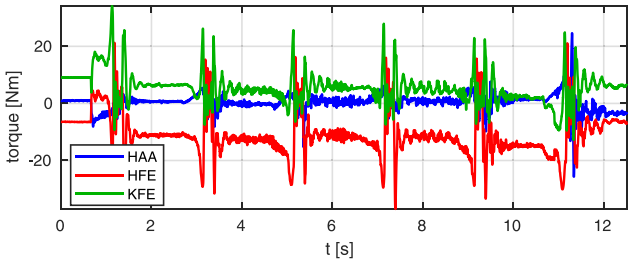

Fig. 6. Joint torques of ANYmal during a repeated squat jump. The torques stay well below the physical torque limit of 40 Nm. Furthermore, the torques drop quite abruptly after take off and then peak during the landing phase.

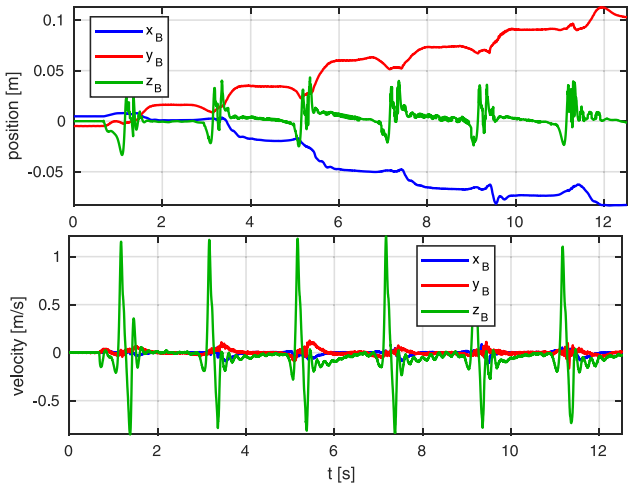

Fig. 7. Base position and linear velocity of ANYmal during a repeated squat jump. We mostly penalize base orientation and linear velocities to obtain straight jump motions. As a result, the controller achieves a constant apex height but drifts slightly in x and y directions. We observe that ANYmal first slightly bents its knees and lowers the base before jumping off to accumulate energy, a behaviour that would not result from a pure feedback controller.

with disturbances. The robot is executing a periodic trot motion in place. At $t = 2.8$ s we placed a wooden plank below one of ANYmal's feet. For the duration of the disturbance the MPC controller escapes the periodic trot and finds different solutions. To mitigate the disturbance the controller shifts and rotates the base and when the disturbance is gone it returns to the periodic trot. Compared to HyQ the deviations of the base orientation and position are much smaller. One of the reasons for this is certainly the size and weight differences between these two robots.

2) Squat Jumps: To underline that the approach can leverage the full system dynamics, we also perform a squat jump. Here, amongst the usual running and final cost, a temporal cost encourages an upward base velocity at a certain time point. However, the lift-off time and especially the landing time are not pre-specified. Also, while it is not surprising that all legs lift-off at the same time, we do not explicitly enforce it. To test repeated jumps, we activate the vertical velocity costs periodically. While the robot does not always land perfectly, the MPC controller optimizes a trajectory from the current state and tries to get back as close as possible to the nominal state. This experiment underlines the robustness of the approach, as several squat jumps can be executed without resetting the robot to the nominal configuration. Figs. 7 and 6 show the measurements of ANYmal during the squat jump experiments. We enforce jumps every

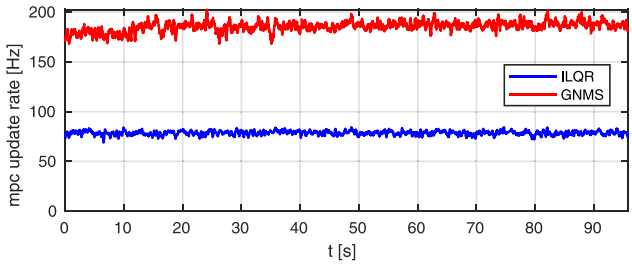

Fig. 8. MPC update rate as recorded during two trotting experiments on ANYmal. While iLQR achieves update rates of around 80 Hz, GNMS reaches almost 190 Hz. While the higher update rate does not lead to significantly better performance, it leaves more headroom for extending the time horizon.

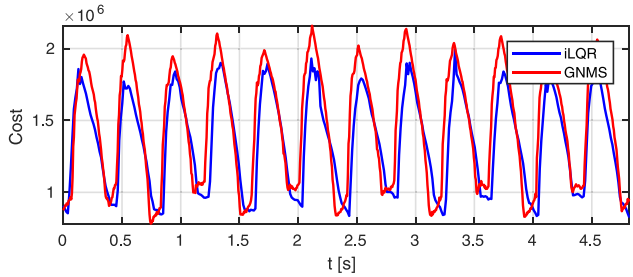

Fig. 9. Total cost of the iLQR and GNMS algorithms during the trot experiment. Both algorithms result in a similar cost after optimization.

2 seconds, starting at $t = 1$ s. The desired take off velocity was 1.0 m/s in vertical direction (z axis) leading to an apex height of 3 cm. The measurements show that the robot slightly overshoots the velocity setpoint by 0.1 m/s during execution. On the torque level the robot stayed well below the admissible torque level of ANYmal (40 Nm) without imposing additional constraints. The robot drifts in x and y direction between consecutive squat jumps. The reason for this is that we do not impose large cost penalties in x and y positions for stability reasons. One interesting observation is that the base height is lowered and the knees are bent before jumping off to accumulate kinetic energy, which reduces the maximum torque used for a given height.

3) Forward Jump: If we add a forward velocity component to the squat jump, we obtain a forward jump behavior. Again, lift-off time and landing time are not pre-specifed. Results of the forward jump can be found in the submitted video.

C. Timings

In this section, we present timings for different solver setups. For all timings and experiments, the NMPC solver is run on an Intel Core i7 4790 quadcore PC with 3.6 GHz.

Fig. 8 shows the timings obtained from the trotting experiments on ANYmal. We measured the rate of incoming trajectories that the tracking controller received. It can be seen that our solver runs at around 80 Hz for when using the iLQR-algorithm and at 175 Hz when using Gauss-Newton Multiple-Shooting. The higher frequency of GNMS-MPC results from three main factors: first, the parallel forward simulation, second, in contrast to iLQR, GNMS does not require a line-search for this particular problem, and third, the GNMS algorithm allows for using a higher control discretization (6 ms). While GNMS offers a higher update rate, it is not very noticeable in performance such that iLQR performs similarly well. This is illustrated in Fig. 9 which compares the costs of the trajectories obtained from both algorithms during a trot task. However, we notice that below 30 Hz update rate, the performance on hardware starts to degrade significantly. Therefore, GNMS leaves more headroom for more complex systems or longer time horizons.

At 80 Hz, a single computation takes less than 12 ms. Compared to [15], which is using the same algorithm but requires around 50 ms per iteration, our solver is about 300% faster for the same time horizon and similar number of degrees of freedom while having a much finer control discretization (4 ms instead of 20 ms) and only using a third of the CPU cores. Assuming the same number of cores and the same discretization, our solver would run about 10–15 times faster than the one presented in [15]. This results from a more efficient solver implementation, optimized vectorization, faster computation of the dynamics due to a simpler contact model as well as using Auto-Diff code-generation. As a result, delays are significantly reduced, which is crucial for robustness.

VII. Summary and Outlook

The presented work shows the first application of whole-body NMPC through contacts for dynamic motions. Furthermore, this is the first time that such an approach is applied to hardware for periodic gait patterns and tasks with dynamic contact switches. We demonstrate that NMPC can be run at rates that exceed the state of the art by an order of magnitude by using an efficient implementation combined with state-of-the-art software engineering techniques such as Auto-Diff, symplectic integration as well as vectorization. By applying our method to two different robots, we show that our integration with state estimation and controllers allows us to deploy the approach to different robots with different actuation principles without major adjustments. The tasks themselves underline the benefits of running an NMPC controller that can reason about contacts. When looking e.g., at the trotting task, we see disturbance-dependent gait pattern changes, base stabilization and small reactive side stepping. If implemented with a classical approach the resulting planning and control pipeline would possible consist of several modules and layers. Using NMPC they emerge naturally from a single algorithm. While cost function tuning remains a manual task to achieve optimal performance, new tasks can be simply tuned in a few minutes. This way, we avoid designing a control and planning framework from scratch when the task at hand changes, which is still often the case in robotics.

While these first results look promising, there is still a significant amount of future work to be considered, both from an algorithmic as well as from an implementation point of view. On the algorithmic side, we plan to leverage the speed advantage and better warm starting capabilities of GNMS to extend the time horizon of the NMPC controller. As part of this work, it would be worthwhile to see how a larger time horizon influences performance and robustness. We expect that a longer time horizon could show more elaborate disturbance rejection and recovery behavior since it offers more flexibility and predictive capabilities to the solver. Furthermore, while most tasks by design stayed within the physical limitations of the platforms, GNMS would allow us to handle constraints such as torque

limitations explicitly as as inequality constraints. Given that the higher update rate of GNMS-MPC did not lead to an increase in performance, we believe that our method can cope with the extra computational complexity when dealing with constraints, especially since the algorithm's complexity remains linear in time. Also, we could resort to soft-constraints rather than hard constraints [40]. From a practical perspective, we wish to include some model adaptation or disturbance estimation to our implementation that robustifies our approach even further. Additionally, this could help to bring new motions to hardware. Finally, the NMPC controller could also be used to track and stabilize motions generated from a high level foothold and motion planner to handle non-convex terrains.

ACKNOWLEDGMENT

We wish to thank the entire SYSCOP group at IMTEK at the University of Freiburg, especially Prof. Moritz Diehl and Dimitris Kouzoupis for their valuable feedback and Gianluca Frison for his support for interfacing with the HPIPM solver. Furthermore, our gratitude goes to Marko Bjelonic and Péter Fankhauser who helped with the experiments.

REFERENCES

[1] A. W. Winkler, C. Mastalli, I. Havoutis, M. Focchi, D. G. Caldwell, and C. Semini, "Planning and execution of dynamic whole-body locomotion for a hydraulic quadruped on challenging terrain," in *Proc. IEEE Int. Conf. Robot. Autom.*, 2015, pp. 5148–5154.

[2] M. Kalakrishnan, J. Buchli, P. Pastor, M. Mistry, and S. Schaal, "Learning, planning, and control for quadruped locomotion over challenging terrain," *Int. J. Robot. Res.*, vol. 30, no. 2, pp. 236–258, 2011.

[3] C. Gehring *et al.*, "Practice makes perfect: An optimization-based approach to controlling agile motions for a quadruped robot," *IEEE Robot. Autom. Mag.*, vol. 23, no. 1, pp. 34–43, Mar. 2016.

[4] C. Mastalli, I. Havoutis, M. Focchi, D. G. Caldwell, and C. Semini, "Hierarchical planning of dynamic movements without scheduled contact sequences," in *Proc. IEEE Int. Conf. Robot. Autom.*, 2016, pp. 4636–4641.

[5] H. Dai, A. Valenzuela, and R. Tedrake, "Whole-body motion planning with centroidal dynamics and full kinematics," in *Proc. IEEE-RAS Int. Conf. Humanoid Robots*, 2015, pp. 295–302.

[6] S. Kuindersma *et al.*, "Optimization-based locomotion planning, estimation, and control design for the atlas humanoid robot," *Auton. Robots*, vol. 40, no. 3, pp. 429–455, 2016.

[7] K. Koyanagi *et al.*, "A pattern generator of humanoid robots walking on a rough terrain using a handrail," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst.*, 2008, pp. 2617–2622.

[8] F. Farshidian, M. Neunert, A. W. Winkler, G. Rey, and J. Buchli, "An efficient optimal planning and control framework for quadrupedal locomotion," in *Proc. IEEE Int. Conf. Robot. Autom.*, 2017, pp. 93–100.

[9] A. Herzog, N. Rotella, S. Schaal, and L. Righetti, "Trajectory generation for multi-contact momentum-control," in *Proc. IEEE-RAS Int. Conf. Humanoid Robots*, 2015, pp. 874–880.

[10] K. Mombaur, "Using optimization to create self-stable human-like running," *Robotica*, vol. 27, no. 03, pp. 321–330, 2009.

[11] M. Posa, C. Cantu, and R. Tedrake, "A direct method for trajectory optimization of rigid bodies through contact," *Int. J. Robot. Res.*, vol. 33, no. 1, pp. 69–81, 2014.

[12] Y. Tassa, T. Erez, and E. Todorov, "Synthesis of robust behaviors through online trajectory optimization," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst.*, 2012, pp. 4906–4913.

[13] D. Pardo, M. Neunert, A. Winkler, R. Grandia, and J. Buchli, "Hybrid direct collocation and control in the constraint-consistent subspace for dynamic legged robot locomotion," *Proc. Robotics: Sci. Syst.*, Cambridge, Massachusetts, Jul. 2017, doi: 10.15607/RSS.2017.XIII.042.

[14] T. Erez, K. Lowrey, Y. Tassa, V. Kumar, S. Kolev, and E. Todorov, "An integrated system for real-time model predictive control of humanoid robots," in *Proc. IEEE-RAS Int. Conf. Humanoid Robots*, 2013, pp. 292–299.

[15] J. Koenemann *et al.*, "Whole-body model-predictive control applied to the HRP-2 humanoid," in *Proc. 2015 IEEE/RSJ Int. Conf. Intell. Robots Syst.*, 2015, pp. 3346–3351.

[16] M. Giftthaler, M. Neunert, M. Stäuble, and J. Buchli, "The control toolbox—an open-source C++ library for robotics, optimal and model predictive control," arXiv:1801.04290 [cs.RO], 2018. [Online]: Available: https://adrlab.bitbucket.io/ct

[17] M. Neunert, F. Farshidian, A. W. Winkler, and J. Buchli, "Trajectory optimization through contacts and automatic gait discovery for quadrupeds," *IEEE Robot. Autom. Lett.*, vol. 2, no. 3, pp. 1502–1509, Jul. 2017.

[18] M. Posa, C. Cantu, and R. Tedrake, "A direct method for trajectory optimization of rigid bodies through contact," *Int. J. Robot. Res.*, vol. 33, no. 1, pp. 69–81, Jan. 2014.

[19] D. Ralph and S. J. Wright, "Some properties of regularization and penalization schemes for MPECs," *Optim. Methods Softw.*, vol. 19, no. 5, pp. 527–556, 2004.

[20] G. Bouza and G. Still, "Mathematical programs with complementarity constraints: convergence properties of a smoothing method," *Math. Operations Res.*, vol. 32, no. 2, pp. 467–483, 2007.

[21] M. Azad and R. Featherstone, "A new nonlinear model of contact normal force," *IEEE Trans. Robot.*, vol. 30, no. 3, pp. 736–739, Jun. 2014.

[22] M. Giftthaler, M. Neunert, M. Stäuble, M. Frigerio, C. Semini, and J. Buchli, "Automatic differentiation of rigid body dynamics for optimal control and estimation," *Adv. Robot.*, vol. 31, pp. 1225–1237, 2017.

[23] M. Diehl, H. G. Bock, H. Diedam, and P.-B. Wieber, "Fast direct multiple shooting algorithms for optimal robot control," in *Fast Motions in Biomechanics and Robotics*. New York, NY, USA: Springer, 2006, pp. 65–93.

[24] M. Posa, S. Kuindersma, and R. Tedrake, "Optimization and stabilization of trajectories for constrained dynamical systems," in *Proc. IEEE Int. Conf. Robot. Autom.*, 2016, pp. 1366–1373.

[25] D. Pardo, L. Möller, M. Neunert, A. W. Winkler, and J. Buchli, "Evaluating direct transcription and nonlinear optimization methods for robot motion planning," *IEEE Robot. Autom. Lett.*, vol. 1, no. 2, pp. 946–953, Jul. 2016.

[26] M. Giftthaler, M. Neunert, M. Stäuble, J. Buchli, and M. Diehl, "A family of iterative Gauss–Newton shooting methods for nonlinear optimal control," *CoRR*, Dec. 2017. [Online]. Available: http://arxiv.org/abs/1711.11006

[27] E. Todorov and W. Li, "A generalized iterative LQG method for locally-optimal feedback control of constrained nonlinear stochastic systems," in *Proc. IEEE Amer. Control Conf.*, 2005, pp. 300–306.

[28] A. Sideris and J. E. Bobrow, "An efficient sequential linear quadratic algorithm for solving nonlinear optimal control problems," *Trans. Autom. Control*, vol. 50, no. 12, pp. 2043–2047, Dec. 2005.

[29] M. Diehl, H. G. Bock, and J. P. Schlöder, "A real-time iteration scheme for nonlinear optimization in optimal feedback control," *SIAM J. Control Optim.*, vol. 43, no. 5, pp. 1714–1736, 2005.

[30] M. Frigerio, J. Buchli, and D. Caldwell, "Code generation of algebraic quantities for robot controllers," in *Proc. 2012 IEEE/RSJ Int. Conf. Intell. Robots Syst.*, Oct. 2012, pp. 2346–2351.

[31] E. Catto, "Soft Constraints – Reinventing the Spring," in *Proc. Game Develop. Conf.*, 2011. [Online]. Available: http://box2d.org/files/GDC2011/GDC2011_Catto_Erin_Soft_Constraints.pdf

[32] M. Diehl and S. Gros, *"Numerical Optimal Control*," to be published.

[33] A. Prabhakar, K. Flakamp, and T. D. Murphey, "Symplectic integration for optimal Ergodic control," in *Proc. 2015 54th IEEE Conf. Decision Control*, Dec. 2015, pp. 2594–2600.

[34] N. Firasta, M. Buxton, P. Jinbo, K. Nasri, and S. Kuo, "Intel AVX: New frontiers in performance improvements and energy efficiency," Intel White Paper, vol. 19, p. 20, 2008. [Online]. Available: http://software.intel.com

[35] G. Guennebaud *et al.*, "Eigen v3." [Online]. Available: http://eigen.tuxfamily.org

[36] C. Semini *et al.*, "Design of HyQ—A hydraulically and electrically actuated quadruped robot," *Inst. Mech. Eng., J. Syst. Control Eng.*, vol. 225, pp. 831–849, 2011.

[37] M. Hutter *et al.*, "Anymal—A highly mobile and dynamic quadrupedal robot," in *Proc. 2016 IEEE/RSJ Int. Conf. Intell. Robots Syst.*, Oct. 2016, pp. 38–44.

[38] J. Pratt, C.-M. Chew, A. Torres, P. Dilworth, and G. Pratt, "Virtual model control: An intuitive approach for bipedal locomotion," *Int. J. Robot. Res.*, vol. 20, no. 2, pp. 129–143, 2001.

[39] M. Bloesch *et al.*, "State estimation for legged robots-consistent fusion of leg kinematics and IMU," *RSS Robot. Sci. Syst.*, vol. 8, pp. 17–24, 2012.

[40] M. Neunert, F. Farshidian, and J. Buchli, "Efficient whole-body trajectory optimization using contact constraint relaxation," in *Proc. IEEE-RAS Int. Conf. Humanoid Robots*, 2016, pp. 43–48.