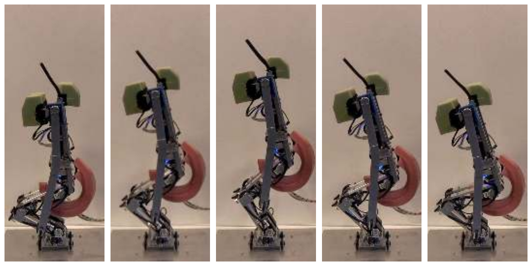
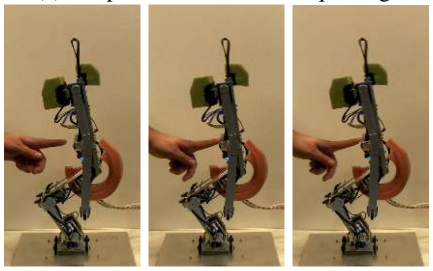
Combining Multi-Level Real-time Iterations of Nonlinear Model Predictive Control to Realize Squatting Motions on Leo

Manuel Kudruss[1,3] Ivan Koryakovskiy[2] Heike Vallery[2] Katja Mombaur[1] Christian Kirches[3]

***Abstract*— Today's humanoid robots are complex mechanical systems with many degrees of freedom that are built to achieve locomotion skills comparable to humans. In order to synthesize whole-body motions, real-tme capable direct methods of optimal control are a subject of contemporary research. To this end, Nonlinear Model Predictive Control is the method of choice to realize motions on the physical robot using model-based optimal control. However, the complexity of the problem results in a high computational time that falls short of the expectations of robotic experimenters and control engineers. In this article, we show how advanced NMPC methods can be applied to improve the control rate by a factor of 10–16 up to 190 Hz. This is achieved by thread-based parallelization of two controllers and by efficiently reusing control problem linearizations of the last iteration to provide fast feedback by one controller while the other controller prepares the next nonlinear step including the evaluation of the multi-body dynamics and the respective sensitivities. This way, the bottleneck of the roll-out of up to 130 ms can partly be side-stepped by repeated calls of the much faster feedback phase of $\sim$ 5 ms. This enables a realization of a squatting task on the actual 2D-robot Leo of Delft University of Technology, which was not possible using a conventional Nonlinear Model Predictive Control scheme.**



(a) Snapshot series of Leo squatting.



(b) Snapshot series of external perturbation.

Fig. 1: Snapshot series of the performed experiments.

I. INTRODUCTION

Today's humanoid robots are complex mechanical systems with many degrees of freedom, a multitude of sensors, on-board computational power and powerful batteries in order to enable autonomy. While they are built to achieve human-like locomotion skills, they still lack online control strategies that could realize these skills. Literature proposes different strategies to synthesize whole-body motions for a computational model of a robotic platform. Methods from the domains of both model-based and model-free optimal control are subject of current research. See [1, 2] as an example of model-free approach to motion generation on the robot.

From the model-based point of view, the method of choice to realize motions on a robotic platform is Nonlinear Model Predictive Control (NMPC). NMPC is a closed-loop control strategy that solves an optimal control problem online, and employs a dynamic model to predict the process on a finite horizon and to compute an optimal control profile. While originally developed for the control of slow chemical processes, recent advances allow the application of NMPC to faster systems. However, the hybrid and nonsmooth nature of multi-body dynamics as well as the high computational complexity still limit the impact of NMPC for motion generation on a much wider scale, and most often falls short of the expectations of robotic experimenters. Therefore, only few realizations of NMPC for motion generation on robotic platforms exist, e.g. [3] using Differential Dynamic Programming, and far less were used to generate walking motions, e.g. [4, 5, 6], where a combination of simplified models and hierarchical control of the robot was used.

Advanced methods of NMPC follow a direct approach to optimal control. In doing so, they first discretize the optimal control problem and then optimize the resulting structured problem by efficient and tailored methods, e.g. using direct multiple shooting [7] or direct collocation [8]. In order to provide control feedback in real time, the idea of real-time iterations for NMPC was developed [9], which is based on the observation that a single NEWTON step for the discretized problem already provides a feedback control that is optimal to first order. Furthermore, the NEWTON step can be separated in three distinct phases, where the time critical feedback phase involves the solution of only a single Quadratic Program (QP) and can be solved effi-

[1] M. Kudruss and K. Mombaur are with the Interdisciplinary Center for Scientific Computing (IWR), Heidelberg University, Im Neuenheimer Feld 205, 69120 Heidelberg, Germany

[2] I. Koryakovskiy and H. Vallery are with the Delft Biorobotics Lab, Faculty of Mechanical Engineering, Delft University of Technology, 2628 CD Delft, The Netherlands.

[3] M. Kudruss and C. Kirches are with the Institute for Mathematical Optimization, Technische Universität Carolo-Wilhelmina zu Braunschweig Universitätsplatz 2, 38106 Braunschweig, Germany.

The research leading to these results has received funding from the European Union Seventh Framework Programme (FP7/2007–2013) under grant agreement n. 611909 (KoroiBot), `www.koroibot.eu`

ciently, cf. [10]. The time-consuming evaluation of the multi-body system dynamics and the respective sensitivities of the preparation phase happens prior to observation of the current system state. The idea of real-time iterations was extended in [11, 12, 13] to include four distinct levels of linerization updates reducing the preparation time. [14] proposes to mix or partially execute the updates. Some aspects of this methodology have also been transferred to interior point approaches, cf. [15].

A. *Contribution of this Article*

For the platform considered in this article, the analysis of the different phases of a NMPC iteration reveals that the bottleneck of computation is the preparation phase, i.e., the forward simulation of the multi-body dynamics and the evaluation of derivative information, while the feedback is reasonably fast. Therefore, in this article, we propose a control strategy based on switching of two controllers. One controller provides feedback based on linear MPC for the last known linearization of the nonlinear control problem. The other controller is given actual initial values and computes a new linearization of the problem. When the problem is successfully linearized, a control feedback based on NMPC is provided and the linearization is communicated to the linear MPC controller. Subsequently, feedback is provided again using Linear Model Predictive Control (LMPC) based on the this new linearization while the other controller is busy relinearizing the problem. The contribution of this article is a new and beneficial combination and parallelization of both linear and nonlinear methods for feedback control.

The proposed strategy is a key to running NMPC on the robotic platform in real time and the performance of the proposed control strategy is evaluated on the 2D robot Leo of Delft University of Technology, cf. [1]. The performed task, squatting of the robotic platform, is realized via a tracking of switching setpoints of the NMPC controller.

II. Nonlinear Model Predictive Control

In this article, feedback is provided by NMPC, a closed-loop control strategy in which the control action is computed from the current system state by solving an open-loop Optimal Control Problem (OCP) on a finite prediction horizon $\mathcal{T} := [0, T]$ online, also denoted as receding horizon control. The OCP to be solved to provide feedback reads

$$\begin{aligned}
\min_{\boldsymbol{x}(\cdot),\boldsymbol{u}(\cdot),\boldsymbol{p}} \quad & \int_0^T \|\boldsymbol{\ell}(\boldsymbol{x}(t),\boldsymbol{u}(t),\boldsymbol{p}) - \bar{\boldsymbol{\ell}}(t,\boldsymbol{p})\|_W^2 \, \mathrm{d}t && \text{(1a)}\\
& + \|e(\boldsymbol{x}(T),\boldsymbol{p}) - \bar{e}(\boldsymbol{p})\|_{\tilde{W}}^2 && \text{(1b)}\\
\text{s.t.} \quad & \dot{\boldsymbol{x}}(t) = \boldsymbol{f}(\boldsymbol{x}(t),\boldsymbol{u}(t),\boldsymbol{p}),\ t \in \mathcal{T}, && \text{(1c)}\\
& 0 = \boldsymbol{x}(0) - \hat{\boldsymbol{x}}_0, && \text{(1d)}\\
& 0 = \boldsymbol{p} - \hat{\boldsymbol{p}}, && \text{(1e)}\\
& 0 \leqslant \boldsymbol{g}(\boldsymbol{x}(t),\boldsymbol{u}(t)), \quad t \in \mathcal{T}. && \text{(1f)}
\end{aligned}$$

For $t \in \mathcal{T}$, $\boldsymbol{u}(t) \in \mathbb{R}^{n_{\boldsymbol{u}}}$ denotes the control trajectory. The state trajectory of the dynamic system is $\boldsymbol{x}(t) \in \mathbb{R}^{n_{\boldsymbol{x}}}$. The model parameters of the system are denoted by $\boldsymbol{p} \in \mathbb{R}^{n_{\boldsymbol{p}}}$. Here, the dynamic system is described by ordinary differential equations with right-hand side $\boldsymbol{f}(\boldsymbol{x},\boldsymbol{u},\boldsymbol{p})$. The nonlinear least-squares objective function defined by a weighted L^2-norm with positive definite weighting matrices W and $\tilde{W}$ is composed of two terms that penalize the deviation of $\boldsymbol{\ell}(\boldsymbol{x},\boldsymbol{u},\boldsymbol{p})$, $e(\boldsymbol{x},\boldsymbol{p})$ from setpoints $\bar{\boldsymbol{\ell}}$, $\bar{e}$. In addition, mixed state-control path constraints $\boldsymbol{g}(\boldsymbol{x},\boldsymbol{u})$ are imposed on the system to model its physical limitations or security margins.

At the current time instant $t = 0$, given the full state $\hat{\boldsymbol{x}}_0 \in \mathbb{R}^{n_{\boldsymbol{x}}}$ and parameter estimates $\hat{\boldsymbol{p}} \in \mathbb{R}^{n_{\boldsymbol{p}}}$, an NMPC scheme solves the open-loop optimal control problem (1) to provide the first part of the approximated optimal solution $\boldsymbol{u}^\star(t;\hat{\boldsymbol{x}}_0,\hat{\boldsymbol{p}}), t \in [t_0, t_1]$ as a feedback control to the system.

A. *Direct Optimal Control*

In order to solve the infinite dimensional problem (1), we apply a direct and all-at-once approach by subdividing the horizon $\mathcal{T}$ into N subintervals $[t_i, t_{i+1}], 0 \leqslant i \leqslant N$, and discretize the control trajectory by means of constant control parameters on the time grid, i.e., $\boldsymbol{u}(t) = \boldsymbol{q}_i, t \in [t_i, t_{i+1}]$.

Direct multiple shooting [7] introduces additional variables $\boldsymbol{s}_i \in \mathbb{R}^{n_{\boldsymbol{x}}}, 0 \leqslant i < N$ and further parametrizes the state trajectory by means of the solution $\boldsymbol{x}(t) := \boldsymbol{x}(t; t_i, \boldsymbol{s}_i, \boldsymbol{q}_i, \boldsymbol{p})$ of separate local initial value problems

$$\dot{\boldsymbol{x}}(t) = \boldsymbol{f}(\boldsymbol{x}(t),\boldsymbol{u}(t),\boldsymbol{p}), \quad \boldsymbol{x}(t_i) = \boldsymbol{s}_i, \quad t \in [t_i, t_{i+1}],$$

by state-of-the-art adaptive solvers for both evaluation of nominal solution as well as sensitivities, e.g. [16]. Continuity of the trajectories in the solution is established by matching conditions $\boldsymbol{x}(t_{i+1}; t_i, \boldsymbol{s}_i, \boldsymbol{q}_i, \boldsymbol{p}) = \boldsymbol{s}_{i+1}, 0 \leqslant i < N$.

From this discretization and parametrization, a large but structured nonlinear programming problem is obtained that can be solved efficiently with tailored structure-exploiting Sequential Quadratic Programming (SQP) methods, cf. [17]. Observing that the OCP is linearly dependent on $\hat{\boldsymbol{x}}_0$, $\hat{\boldsymbol{p}}$ due to (1d),(1e), the Nonlinear Program (NLP) may be written in parametric form,

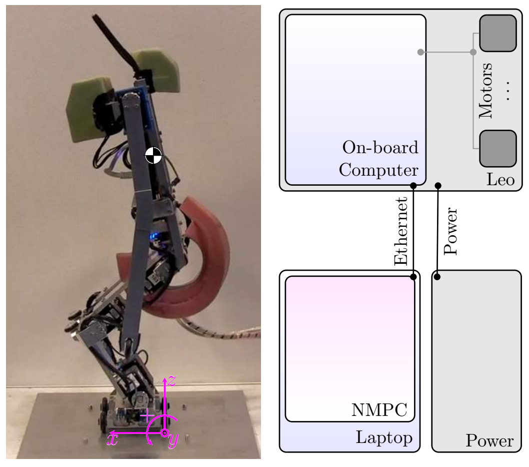
$$\begin{aligned}
\min_{\boldsymbol{w}} \quad & \varphi(\boldsymbol{w}) && \text{(2a)}\\
\text{s.t.} \quad & 0 = \boldsymbol{c}(\boldsymbol{w}) + \boldsymbol{P}(\hat{\boldsymbol{x}}_0, \hat{\boldsymbol{p}}), && \text{(2b)}\\
& 0 \leqslant \boldsymbol{d}(\boldsymbol{w}), && \text{(2c)}
\end{aligned}$$

with $\boldsymbol{w} := [\boldsymbol{s}, \boldsymbol{q}, \boldsymbol{p}]$, $\boldsymbol{s} := [\boldsymbol{s}_0, \ldots, \boldsymbol{s}_N]$, $\boldsymbol{q} := [\boldsymbol{q}_0, \ldots, \boldsymbol{q}_N]$. The projector $\boldsymbol{P}$ aligns the initial values and parameter with the rest of the equality constraints $\boldsymbol{c}$.

Starting with an initial guess $(\boldsymbol{w}^0, \boldsymbol{\lambda}^0, \boldsymbol{\mu}^0)$ of the primal and dual variables of NLP (2), a full-step SQP method employs, in every iteration k, a quadratic approximation of NLP in the form of a QP and performs a step $\boldsymbol{w}^{k+1} = \boldsymbol{w}^k + \Delta\boldsymbol{w}^k, \boldsymbol{\lambda}^{k+1} = \boldsymbol{\lambda}_{QP}, \boldsymbol{\mu}^{k+1} = \boldsymbol{\mu}_{QP}$ by using the solution $(\Delta\boldsymbol{w}^0, \boldsymbol{\lambda}_{QP}, \boldsymbol{\mu}_{QP})$ of the QP, which is given by

$$\begin{aligned}
\min_{\Delta\boldsymbol{w}} \quad & \tfrac{1}{2}\Delta\boldsymbol{w}^T \boldsymbol{B}(\boldsymbol{w}^k)\Delta\boldsymbol{w} + \Delta\boldsymbol{w}^T \boldsymbol{b}(\boldsymbol{w}^k) && \text{(3a)}\\
\text{s.t.} \quad & 0 = \boldsymbol{c}(\boldsymbol{w}^k) + \boldsymbol{C}(\boldsymbol{w}^k)\Delta\boldsymbol{w} + \boldsymbol{P}(\hat{\boldsymbol{x}}_0, \hat{\boldsymbol{p}}), && \text{(3b)}\\
& 0 \leqslant \boldsymbol{d}(\boldsymbol{w}^k) + \boldsymbol{D}(\boldsymbol{w}^k)\Delta\boldsymbol{w}, && \text{(3c)}
\end{aligned}$$

Here, we denote the Jacobians of the in- and equality constraints by $\boldsymbol{C} = \frac{\mathrm{d}\boldsymbol{c}}{\mathrm{d}\boldsymbol{w}}(\boldsymbol{w})$, $\boldsymbol{D} = \frac{\mathrm{d}\boldsymbol{d}}{\mathrm{d}\boldsymbol{w}}(\boldsymbol{w})$, respectively. The



(a) (b)

Fig. 2: Leo robot of Delft University of Technology (a) and its control architecture (b). Annotation shows the world frame (magenta) as well as the root frame origin (black).

Hessian of the Lagrangian of NLP is $\boldsymbol{B}(\boldsymbol{w})$, and the vector $\boldsymbol{b}(\boldsymbol{w})$ is the gradient of the objective.

State-of-the-art NMPC methods based on nonlinear programming rely on the real-time iteration scheme of [9] to compute feedback in real-time. Subsequent problems only differ in the values of $\hat{\boldsymbol{x}}_0$, $\hat{\boldsymbol{p}}$ and this dependency is only linear in the QP. Due to the linearity, by means of the so-called *initial value embedding* methodology, i.e., a difference in $\boldsymbol{s}_0 \neq \hat{\boldsymbol{x}}_0$ and $\boldsymbol{p} \neq \hat{\boldsymbol{p}}$ due to initializing the problem with the state and control information of the last solution can be satisfied after a single full NEWTON step. The next iterate then represents a first-order tangential predictor of the solution, cf. [18].

In this way, computationally expensive parts can be separated from time-critical ones and the computational delay of the feedback is reduced to the time required to solve a single QP, such that by careful initialization a separation into three phases of the SQP step is possible as follows:

- **Preparation:** setup of QP (3) and structure exploitation
- **Feedback:** QP solution triggered by arrival of $\hat{\boldsymbol{x}}_0, \hat{\boldsymbol{p}}$
- **Transition:** unroll exploitation, perform NEWTON step

Advanced methods further these ideas by dividing the real-time iteration into sub steps that can provide feedback even faster by evaluating only parts of the required Jacobian information, c.f. [11, 12, 13, 14].

III. ROBOT AND MODEL

In Figure 2, the robot Leo is depicted in (a) as well as its control structure in (b). Leo was originally built to perform Reinforcement Learning experiments directly on the hardware, cf. [1]. The robot is attached to a boom for walking experiments to enforce a 2D motion around a center platform. For the experiments in this article and to further investigate the combination of NMPC and Reinforcement Learning, we focus on a more secure and easier repeatable task than walking, namely squatting. Therefore, the robot was taken off the boom and fixed to a ground plate below its feet in order to ready it for a squatting task. Leo is small ($50\,\mathrm{cm}$) in size and light-weight ($1.7\,\mathrm{kg}$). Foam bumpers are attached on both sides of the torso top and between the hip motors in order to secure the robot from damage due to falls in a wide range of configurations. Additionally, elastic couplings were added for protection of in-build motor gearboxes. Leo has seven Degrees of Freedom (DoFs) driven by servo motors (Dynamixel XM430; max. torque $3\,\mathrm{N\,m}$), three in each leg at ankle, knee and hip as well as one motor in the shoulder joint. The on-board embedded computer (VIA Eden 1.2GHz CPU and 1GB RAM) communicates over RS-485 serial ports with the motors, which are used in voltage control mode, and communicate back their position as well as temperature. The measured joint velocities are retrieved from the position signal by differentiating and filtering the results by means of a second order BUTTERWORTH filter. The operating system on-board of the robot implements a fixed sampling time of $30\,\mathrm{ms}$. While in principle its possible to run NMPC on the embedded hardware, we choose to use external computational power to implement the control for the robot. As depicted in Figure 2 (b) NMPC is run on a laptop that communicates with the embedded system on the robot via Ethernet using the *ZeroMQ*[1] library for communication. The external computational power was provided by a laptop equipped with an *Intel*© *Core*™*i7 4600U* CPU running at $2.10\,\mathrm{GHz}$ with 4 cores, $8\,\mathrm{GB}$ main memory and running 64–bit *Ubuntu*© *Linux*™.

A. Model

The robot is best approximated by a rigid-body dynamics model. The required dynamic properties of the links, i.e., masses, inertias and Centers of Mass (CoMs), were taken from [1]. We use the software library Rigid Body Dynamics Library (RBDL) [19, 20] to implement the model and to evaluate the Forward Dynamics of the model efficiently.

In Figure 2, the world frame is depicted, where the x axis exists along the 2D walking motion pointing forward, the z axis point upward and, in order to have a CARTESIAN frame, the y axis is pointing out of the figure such that a positive rotation around the y axis results in a clock-wise motion. The kinematic structure of the robot can be modeled in different ways. Here, we employ a fixed-base model that describes the kinematic chain from foot over ankle, knee and hip to the root and the arm of the robot. Furthermore, it is possible to exploit the symmetry for the task as in order to achieve squatting the legs and feet have to be aligned in parallel.

a) States and Controls: The dynamic model is implemented as Ordinary Differential Equation (ODE) in the OCP (1c) by order reduction, i.e., implemented on acceleration level by evaluating the Forward Dynamics (FD), $\ddot{\boldsymbol{q}} = \mathrm{FD}(\boldsymbol{q}, \dot{\boldsymbol{q}}, \boldsymbol{\tau})$, and integrating twice to receive joint positions and velocities $\boldsymbol{q}$, $\dot{\boldsymbol{q}}$.

Following the formulation of the NMPC problem (1), the differential states $\boldsymbol{x}$ are in this case the joint positions $\boldsymbol{q}$ and

[1]http://zeromq.org

velocities $\dot{\boldsymbol{q}}$, i.e., $\boldsymbol{x}(t) = \begin{bmatrix} \boldsymbol{q}(t) \\ \dot{\boldsymbol{q}}(t) \end{bmatrix}$, $\forall t \in \mathcal{T}$.

The controllable variables of the system $\boldsymbol{u}$ are the input voltages of the servo motors $\boldsymbol{v}$, i.e., $\boldsymbol{u}(t) = \boldsymbol{v}(t)$, $\forall t \in \mathcal{T}$. Control functions $\boldsymbol{u}$ are approximated as piecewise constant functions. The voltages $\boldsymbol{v}$ are setpoints provided to the motor internal controller. In order to receive joint torques from the motor input voltage $\boldsymbol{v}$ for the model, we apply a mapping depending on the angular velocity $\dot{\boldsymbol{q}}$, i.e., $\boldsymbol{\tau} \equiv \boldsymbol{\tau}(\boldsymbol{v}, \dot{\boldsymbol{q}})$. We refer to [1] for further details.

b) Objective Function: As mentioned above, the squatting task is realized by means of the objective function

$$\boldsymbol{\ell}(\boldsymbol{x}, \boldsymbol{u}, \boldsymbol{p}) = \begin{bmatrix} \boldsymbol{r}_z(\boldsymbol{q}) \\ x_c(\boldsymbol{q}) \\ pose(\boldsymbol{q}) \\ \dot{\boldsymbol{q}} \end{bmatrix}, \quad \bar{\boldsymbol{\ell}}(t, \boldsymbol{p}) = \begin{bmatrix} \boldsymbol{p} \\ \bar{x}_c \\ 0.30 \\ \boldsymbol{0} \end{bmatrix}, \tag{4}$$

where we denote the forward kinematic evaluations of the root frame as $\boldsymbol{r}(\boldsymbol{q})$ and the x position of the CoM as $x_c(\boldsymbol{q})$. The term $pose(\boldsymbol{q}) \coloneqq \boldsymbol{q}_{\text{ankle}} + \boldsymbol{q}_{\text{knee}} + \boldsymbol{q}_{\text{hip}}$ denotes the evaluation of the torso angle wrt. the world frame computed from the angles of ankle, knee and hip, and the joint velocities $\dot{\boldsymbol{q}}$ are used as a regularization terms improving the stability of the robot. The setpoint for the CoM term is the center of the support polygon denoted by $\bar{x}_c$ and given by the position of the tip and heel of the support foot. The height to track by the robot is provided via the model parameter $\boldsymbol{p}$ and the squatting is then realized by its modulation over time. We weight the different terms by the weighting matrix $W = \text{diag}\{50.0, 100.0, 50.0, 3.0, \ldots, 3.0\}$, with a focus on stability, a trade-off between tracking and correct upright pose and a small weight on the joint velocity penalty.

c) Constraints: In order to guarantee stability of the robot, we additionally formulate static stability as constraint

$$\boldsymbol{g}(\boldsymbol{x}, \boldsymbol{u}) = \begin{bmatrix} x_t - x_c(\boldsymbol{q}) \\ x_c(\boldsymbol{q}) - x_h \end{bmatrix}, \tag{5}$$

where x_t, x_h denote the position of the tip and the heel of the robot feet. We exploit the symmetry and therefore one foot is enough to describe the polygon of support. The optimization is subject to the constraints

$$\begin{bmatrix} -1.57 \\ -2.53 \\ -0.61 \\ -3.00 \end{bmatrix} \leqslant \boldsymbol{q} \leqslant \begin{bmatrix} 1.45 \\ -0.02 \\ 2.53 \\ 0.36 \end{bmatrix} \tag{6}$$

and box constraints to limit the motor voltages by $-5\,\text{V} \leqslant \boldsymbol{v}_i \leqslant 5\,\text{V}$, $i \in \{\text{ankle}, \text{knee}, \text{hip}, \text{arm}\}$.

d) Run-time Analysis in Simulation: Initially, we performed experiments in simulation. From this, an analysis of the computational time of the different NMPC phases is visualized in Figure 7a by box plots. The results show a relatively slow preparation phase of $71.61\,\text{ms}$, while transition ($7.39\,\text{ms}$) and feedback ($5.37\,\text{ms}$) phases are much faster in comparison. Following this, nominal NMPC with a total iteration time of $80.31\,\text{ms}$ would end up with a minimum control rate of only $\sim 12\,\text{Hz}$, while a control rate of $\sim 186\,\text{Hz}$ would be possible by leveraging the fast feedback time this would improve control rate by a factor of 15. We only compared the maximum recorded run time data here and in the mean higher values are possible.

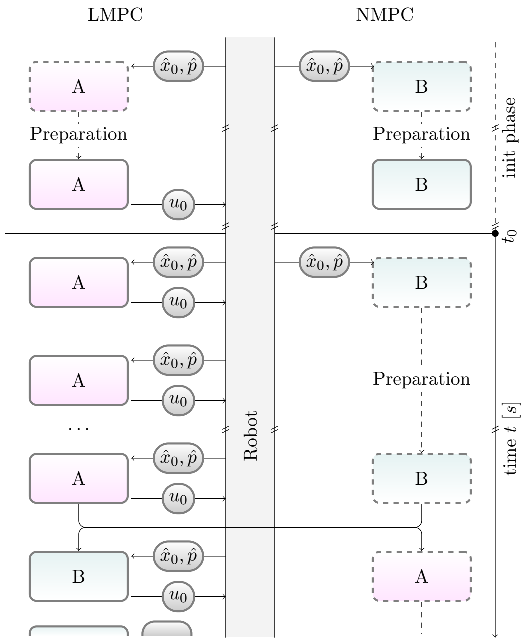


Fig. 3: Schematic of the control approach. In a init(ialization) phase both controllers (A,B) are prepared by estimates ($\hat{\boldsymbol{x}}_0$, $\hat{\boldsymbol{p}}$) from the robot. The concurrent controllers provide feedback $\boldsymbol{u}_0$ either with (LMPC) or re-linearize and provide nonlinear feedback once (NMPC). LMPC feedback is queried as long as NMPC thread is in preparation. When the NMPC thread is ready, roles are switched and the scheme is continued.

IV. Combining Multi-level Real-time Iterations

The idea of different levels of real-time iterations was already touched in Section II. Here, we revisit the levels proposed in [11] that are required for our implementation. In order to provide feedback even faster, the idea of the four different levels is to only update selected parts of the linearization during the preparation phase. This can be understood as updating the respective quantities in the QP (3), i.e., the evaluation of all, parts or none of $\boldsymbol{B}$, $\boldsymbol{b}$, $\boldsymbol{C}$, $\boldsymbol{c}$, $\boldsymbol{D}$ and $\boldsymbol{d}$. We focus on the two extremes of updating the full information and leaving out the preparation phase, i.e., the levels “D” and “A” of [11].

a) Level D: A nominal NMPC iteration, i.e., a full SQP step, evaluating the gradient of the objective $\boldsymbol{b}(\boldsymbol{w}^k)$, the constraint residuals $\boldsymbol{c}(\boldsymbol{w}^k)$, $\boldsymbol{d}(\boldsymbol{w}^k)$ as well as the respective Jacobians $\boldsymbol{C}(\boldsymbol{w}^k)$, $\boldsymbol{D}(\boldsymbol{w}^k)$ and computing a new Hessian (approximation) $\boldsymbol{B}(\boldsymbol{w}^k)$ using the latest iterate $(\boldsymbol{w}^k, \boldsymbol{\lambda}^k, \boldsymbol{\mu}^k)$. From this information a new QP (3) is built. After the solution of the QP is available, the feedback control is sent to the process and the SQP iteration is finalized.

b) Level A: This level realizes LMPC using the most recent linearization provided by a completed D iteration. That is, the most recent set of matrices and vector $\hat{\boldsymbol{B}}$, $\hat{\boldsymbol{b}}$, the constraints $\hat{\boldsymbol{C}}$, $\hat{\boldsymbol{c}}$, $\hat{\boldsymbol{D}}$, $\hat{\boldsymbol{d}}$ is kept fixed in the QP. From this, and given new estimates of the initial states $\hat{\boldsymbol{x}}_0$ as well as parameters $\hat{\boldsymbol{p}}$, a feedback control is computed by solving the QP. Level A iterations can be performed without any evaluation of the nominal or derivative information and consist of only a solution of the already prepared QP.

In Section III, we concluded with the fact that the feedback phase can grant feedback reasonably fast in comparison to the slow preparation phase. This means that feedback can be provided fast and cheap by using only Level A iterations. Level D iterations are computationally expenisve and slightly violate the real-time constraint due the fixed sampling time of the robot. Following the ideas proposed in [14], a solution is to reuse the linearization and provide intermediate feedback using level A, while the preparation phase is ongoing. This can be realized by two concurrent threads either using level A or level D real-time iterations. In this article, we implemented the control scheme as depicted in Figure 3.

First, both controllers are prepared in separate threads and receive initial states $\hat{\boldsymbol{x}}_0$ and parameters $\hat{\boldsymbol{p}}$ in a common initialization phase. While the threads run concurrently, each of the controllers has a distinct role in providing feedback $\boldsymbol{u}_0$ either using level A iterations and reusing the last linearization (LMPC), or in using level D iterations with a full preparation phase (NMPC). The level D iterations are subject to a non-negligible computational delay. Once the relinearization has been triggered by initial values, we provide linear feedback through queries to the level A iterations thread. As soon as the level D thread is ready, which typically happens after 2-3 level A queries here, we switch roles for one iteration, provide full NMPC feedback, and re-linearize in the NMPC thread. The level A thread then takes care of providing the feedback until the next level D iteration is ready.

Software: The approach is implemented on top of OCP solver *MUSCOD-II*, cf. [21, 22]. Here, the preparation phase uses a RUNGE-KUTTA-FEHLBERG $4/5$ method to evaluate the ODEs adaptively, which is extended to also compute the necessary derivative information by means of Internal Numerical Differentiation, cf. [7]. In order to solve the QP in the feedback phase, we employ the QP solver *QPOPT* [23], which itself uses an active-set strategy and implements consecutive warmstarts on the optimal active set of the previous solution. For controlling the robot and investigating the combination with Reinforcement Learning (RL), we integrated *MUSCOD-II* with the RL software package *GRL*[2] [24] that already implements the interface for communicating with robot Leo.

V. EXPERIMENTAL SETUP

In order to evaluate the novel control algorithm, we performed two sets of experiments: a squatting task and an experiment evaluating the performance against perturbations.

The experimental setup of the squatting task is shown in Figure 4. In order to bring the robot into a well-defined

[2]`https://github.com/wcaarls/grl`

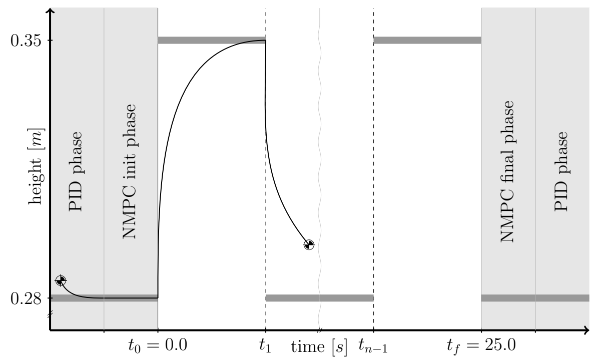


Fig. 4: Scheme explaining the squatting task. Before and after the experiment (white background, from t_0 to t_f), a setup and tear down phase (light gray background) brings the robot into a initial configuration. During the experiment the setpoint (dark gray bars) tracked by the root center (cross shape, cf. Figure 2) is switched at time points $t_i, 0 < i < f$.

initial configuration, two phases precede each trial: a PID phase and an NMPC init(ialization) phase. In the PID phase a configuration is tracked by a low gain PID controller and in the following init phase the NMPC software is initialized. The actual trial starts as soon as the controller is ready, which we indicate by $t_0 = 0.0\,\mathrm{s}$ and ends after $t_f = 25.0\,\mathrm{s}$. The actual task is implemented as a tracking of a setpoint at the center of the torso of the robot by NMPC run on the laptop. The squatting motion is achieved by tracking two distinct setpoints, a lower $\boldsymbol{p}^{lo} = 0.28\,\mathrm{m}$ and $\boldsymbol{p}^{up} = 0.35\,\mathrm{m}$ one, such that the robot has to move its root by $7\,\mathrm{cm}$ from a crouching to almost fully stretched legs. These are repeatedly switched for the controller after periods of $1.5\,\mathrm{s}$, such that 8 full squats have to be performed in total. After the experiment the robot is brought back into its initial configuration by a non-recorded NMPC phase followed by a final PID phase, during which the NMPC policy is finalized.

The setup to evaluate the performance against external perturbations follows that of the squatting task. Here, we track an intermediate setpoint $\boldsymbol{p}^{im} = 0.32\,\mathrm{m}$, while the robot experiences external perturbations.

For the purpose of evaluation the following criteria are considered in the validation of each experiment:

- The value of the objective as indicator of performance.
- The trajectories of states, control and tracking task.
- The timing statistics of the respective NMPC policy.

Friction Compensation for Dynamixel Motors: The actuation of the robot is realized by *Dynamixel XM430* servo motors. Here, we use voltage control mode of the motors, as proposed in [1]. The motors are subject to COULOMB and viscous frictions, and gearbox inefficiency. Without a compensation of the friction a realization of NMPC was not possible on the hardware (see the supplementary video). Therefore, we implemented an affine transformation of an input control signal. The actual applied voltage $\boldsymbol{v}_i = 0.75 \cdot \boldsymbol{v}_i^{nmpc} + \boldsymbol{v}_i^{comp}(\boldsymbol{p})$ is composed of the control signal from NMPC multiplied by the gearbox efficiency of $75\,\%$ and a compensatory term for the friction $\boldsymbol{v}_i^{comp}$, given by

$$\boldsymbol{v}_i^{\mathrm{comp}}(\boldsymbol{p}) = \begin{cases} 1.0\mu_C, & \text{if } \boldsymbol{p} = \boldsymbol{p}^{up} \\ -1.5\mu_C, & \text{if } \boldsymbol{p} = \boldsymbol{p}^{lo}, \end{cases}$$

where the COULOMB friction term was tuned to be $\mu_C = 0.86\,\mathrm{V}$ for $i \in \{\mathrm{ankle}, \mathrm{knee}, \mathrm{hip}\}$ and $0\,\mathrm{V}$ for arm. Please note that methods of system identification can be applied to compensate on the joint level. In the future, we plan to overcome this issue by combining the control and learning of these structural uncertainties in the environment via RL, cf. [25]. Therefore, we omit the parameter identification part in this article and focus on the control.

VI. Results

In Figure 1a, a scene from a squatting experiment shows Leo performing a single squat. The whole experiment was repeated five times in a row. Each trial was successful and no problems occurred. The recorded data from the experiments is visualized in Fig. 5, 6. Note that the joint angles and the motor input voltages of the arm have been dropped. This is due the fact that the arm is not moving much and also has little effect due to its low weight.

In Fig. 5a, the actual height tracking is depicted as presented in Figure 4. The setpoints $p^{\mathrm{up}}, p^{\mathrm{lo}}$ (dark and light gray horizontal bars) attracts the root origin correctly, which then slightly overshoots before the setpoint is switched again. During tracking of the lower setpoint the overshoot is stronger. We use the same colors as for the setpoints $p^{\mathrm{up}}, p^{\mathrm{lo}}$ (dark and light gray) to highlight ascending and descending phases in the other plots.

In Fig. 5b, the reward, i.e., the instantaneous negative objective function value (4), is plotted against time. Here, optimal performance would show values close to zero. While this value is never reached exactly, it comes close to the optimal value at the end of reaching the lower set point. However during each ascending phase the value of the reward is significantly lower than during the respective get-down phase of a squat.

Fig. 6 (a-c), show the angles of hip, knee and ankle joint. All three joints show both a high quantitative similarity of the different trials, with the only visible deviations near the extrema of the trajectories. Looking at the markers, slight deviations in the time profile are visible. While the trajectories are similar, the paths are not perfectly synchronized in time.

Fig. 6 (d-f), show the motor input voltages at hip, knee, and ankle. While they show the same qualitative behavior, one can see the control effort due to the deviations during the tracking of the lower setpoint. Even though the joint angles show satisfactory behavior, the variability in control can also be seen in the video, especially near the end of the experiment when, in the final phase, the lower setpoint is tracked. During ascending the knee voltage saturates to the limit of $5\,\mathrm{V}$ in the very beginning of the motion.

The timing statistics derived from the experiments are shown in Figure 7. Figure 7a reveals similar results for the timing statistics as the numerical experiments. Comparing the maximum values of each phase, we see again a dominant preparation phase of $132.04\,\mathrm{ms}$, which stands out in comparison to the fast feedback and transition phases of $13.32\,\mathrm{ms}$ (and $15.15\,\mathrm{ms}$). This shows again the benefit of the proposed multi-level approach as with this a speedup of the feedback time by a factor of 10–16 is easily possible and one would end up with a minimum control rate of $\sim 75\,\mathrm{Hz}$ and up to $\sim 190\,\mathrm{Hz}$ (mean). In contrast to this, a nominal scheme relying on the same timing statistics would result in a minimum control of not even $\sim 10\,\mathrm{Hz}$.

In Figure 7c, the feedback times for the currently active LMPC thread are shown. While most of the feedback times are below $5\,\mathrm{ms}$, outliers still show feedback times well beyond $10\,\mathrm{ms}$.

In Figure 1b, Leo recovers from the external push performed by an experimenter. The experiment was conducted once. Note that the trajectories of this experiment are not shown due to page limitations. The reader is referred to the supplementary video. The feedback was crisp and the robot directly reacted to the external perturbations by counteracting the force. However, due to the friction compensation the robot started to stretch his legs after some hard perturbations.

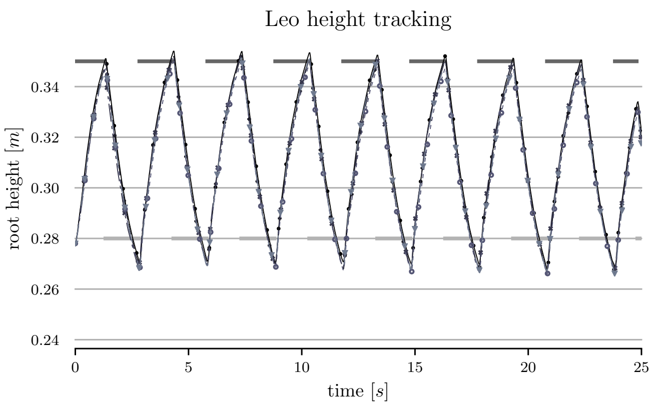
VII. Discussion

The proposed multi-level NMPC control scheme showed a satisfactory real-time performance on the robot. While the friction compensation made the scheme actually work on the hardware, the results reveal the erratic behavior due to model-plant mismatch in the friction model, e.g. overshoot and voltage saturation during squatting and leg stretching in the push recovery experiment. We believe that this is not a problem for future research, as there are avenues to compensating for this either by improving the model using parameter identification methods, or by learning by means of RL, cf. [25].

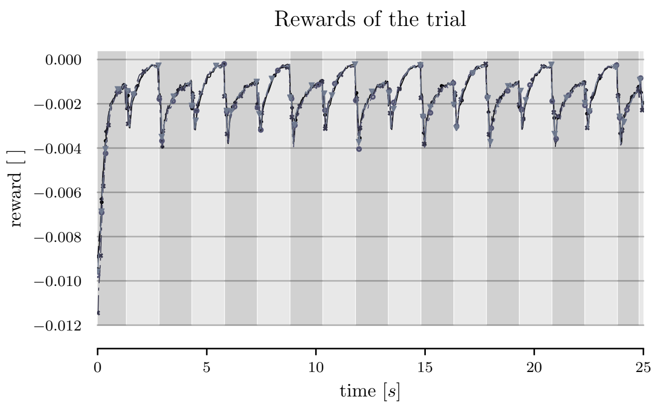
During the descending phase while squatting, or in the push experiment, we noticed a little jittering in the control. We offer two explanations for this. First, the friction compensation alters the control signal computed from the model. Especially during the down-phase or when the robot moves very slowly, we could see the importance of adjusting the compensatory term. Further improving it might help reducing the mismatch or compensating it on motor level will result in smoother motions also in regimes with slow speed. Second, switching the controller also means switching the current linearization point of the model, followed by up to four consecutive feedback phases using it. A change of the linearization has significant impact on the feedback. By further improving the speed of the computations, especially during the preparation phase, both the feedback time as well as the ratio of A and D iterations can be improved.

VIII. Conclusion and Outlook

In this article, we presented a control approach to Nonlinear Model Predictive Control (NMPC) for feedback control employing a combination and parallelization of both linear and nonlinear methods that is based on two different levels of

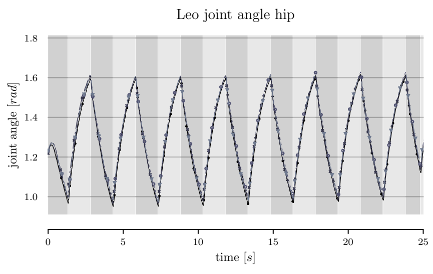


(a)

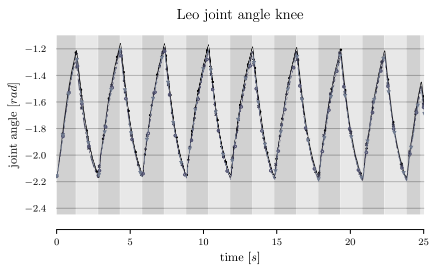


(b)

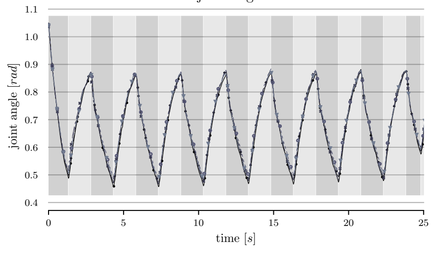
Fig. 5: Recorded and processed data of the squatting experiment: height tracking (a) and instantaneous reward (b).



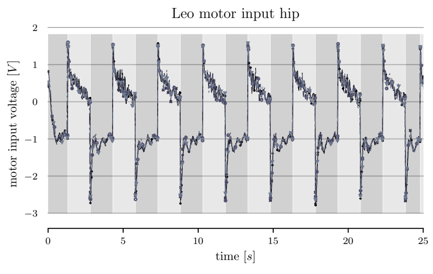
(a)



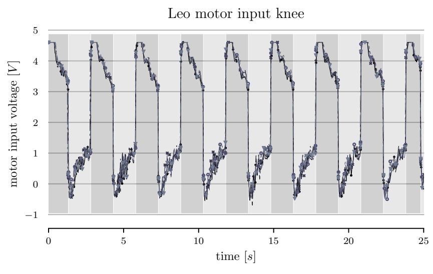
(b)



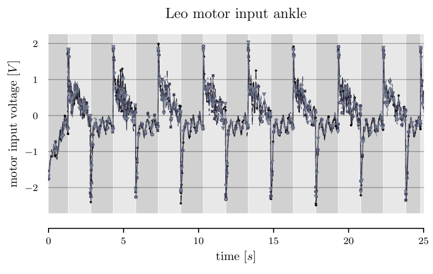
(c)



(d)

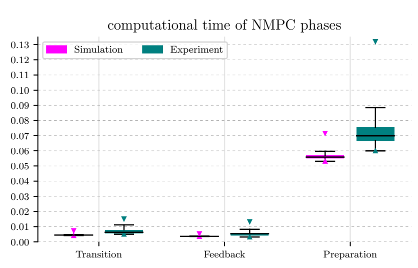


(e)

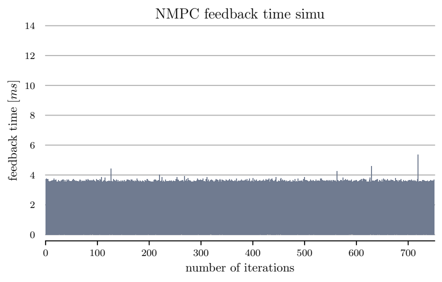


(f)

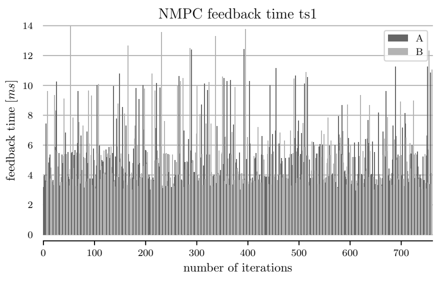
Fig. 6: Recorded and processed data of the squatting experiment: joint angle trajectories (a-c) and motor input voltages (d-f).



(a) Timing of NMPC phases: simulation (lower, magenta), experiment (upper, teal).



(b) Feedback times of nominal NMPC of simulation trial.



(c) Feedback times of thread A, B (magenta, teal) from 1st trial of the experiment.

Fig. 7: Run-time analysis of a squatting experiment under ideal conditions in simulation and of the real experiment.

real-time iterations, named levels A and D. This combination enabled us to achieve real-time execution of NMPC on the physical robotic hardware of Leo. Additionally, we proposed a problem specific model for compensation of the static friction hindering the robot to perform the task. We successfully performed experiments on the robot in different scenarios followed by the detailed analysis of the recorded results. The control scheme showed a convincing performance, where most of the current problems, e.g. computation complexity, were overcome.

This work presents the first step on evaluating the combination of different multi-level real-time iteration of NMPC on the robot, and we plan to further investigate this intriguing combination. Previously reported results from simulation suggest that a performance improvement and a safer exploration are made possible by combining NMPC with Reinforcement Learning on top. After a first successful evaluation on the robot, we plan to focus on more difficult tasks, where fast and robust walking for the platform is the goal.

ACKNOWLEDGMENTS

The authors thank *ROBOTIS* company for providing Dynamixel motors used in experiments with the real robot.

REFERENCES

[1] E. Schuitema, "Reinforcement Learning on autonomous humanoid robots," PhD Thesis, Delft University of Technology, Netherlands, 2012.

[2] I. Mordatch, N. Mishra, C. Eppner, and P. Abbeel, "Combining model-based policy search with online model learning for control of physical humanoids," in *IEEE International Conference on Robotics and Automation (ICRA)*, 2016, pp. 242–248.

[3] J. Koenemann, A. Del Prete, Y. Tassa, E. Todorov, O. Stasse, M. Bennewitz, and N. Mansard, "Whole-body Model-Predictive Control applied to the HRP-2 humanoid," in *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2015.

[4] M. Naveau, M. Kudruss, O. Stasse, C. Kirches, K. Mombaur, and P. Soures, "A reactive walking pattern generator based on nonlinear model predictive control," *IEEE Robotics and Automation Letters*, vol. 2, no. 1, pp. 10–27, 2017.

[5] M. Karklinsky, M. Naveau, A. Mukovskiy, O. Stasse, T. Flash, and P. Soueres, "Robust human-inspired power law trajectories for humanoid HRP-2 robot," in *IEEE International Conference on Biomedical Robotics and Biomechatronics (BioRob)*, 2016, pp. 106–113.

[6] G. D. Magistris, A. Pajon, S. Miossec, and A. Kheddar, "Humanoid walking with compliant soles using a deformation estimator," in *IEEE International Conference on Robotics and Automation (ICRA)*, 2016, pp. 1757–1762.

[7] H. G. Bock and K. J. Plitt, "A Multiple Shooting algorithm for direct solution of optimal control problems," in *Proceedings of the 9th IFAC World Congress*. Pergamon Press, 1984, pp. 242–247.

[8] L. T. Biegler, "Solution of dynamic optimization problems by successive quadratic programming and orthogonal collocation," *Computers & Chemical Engineering*, vol. 8, no. 3, pp. 243–247, 1984.

[9] M. Diehl, *Real-Time Optimization for Large Scale Nonlinear Processes*, ser. Fortschritt-Berichte VDI Reihe 8, Meß-, Steuerungs- und Regelungstechnik, 2002, vol. 920.

[10] H. Ferreau, C. Kirches, A. Potschka, H. Bock, and M. Diehl, "qpOASES: A parametric active-set algorithm for quadratic programming," *Mathematical Programming Computation*, 2014.

[11] H. G. Bock, M. Diehl, E. A. Kostina, and J. P. Schlöder, "Constrained optimal feedback control of systems governed by large differential algebraic equations," in *Real-Time PDE-Constrained Optimization*, L. Biegler, O. Ghattas, M. Heinkenschloss, D. Keyes, and B. van Bloemen Waanders, Eds. SIAM, 2007, ch. 1, pp. 3–24.

[12] J. Albersmeyer, D. Beigel, C. Kirches, L. Wirsching, H. G. Bock, and J. P. Schlöder, "Fast nonlinear model predictive control with an application in automotive engineering," in *Nonlinear Model Predictive Control*, ser. Lecture Notes in Control and Information Sciences, L. Magni, D. Raimondo, and F. Allgöwer, Eds. Springer Berlin Heidelberg, 2009, vol. 384, pp. 471–480.

[13] C. Kirches, L. Wirsching, H. G. Bock, and J. P. Schlöder, "Efficient direct multiple shooting for Nonlinear Model Predictive Control on long horizons," *Journal of Process Control*, vol. 22, no. 3, pp. 540–550, 2012.

[14] J. V. Frasch, L. Wirsching, S. Sager, and H. G. Bock, "Mixed–level iteration schemes for Nonlinear Model Predictive Control," *IFAC Proceedings Volumes*, vol. 45, no. 17, pp. 138–144, 2012.

[15] V. M. Zavala and L. T. Biegler, "The advanced-step NMPC controller: Optimality, stability and robustness," *Automatica*, vol. 45, no. 1, pp. 86–93, 2009.

[16] J. Albersmeyer, "Adjoint based algorithms and numerical methods for sensitivity generation and optimization of large scale dynamic systems," PhD Thesis, Heidelberg University, 2010.

[17] D. B. Leineweber, I. Bauer, H. G. Bock, and J. P. Schlöder, "An efficient multiple shooting based reduced SQP strategy for large-scale dynamic process optimization. Part 1: Theoretical aspects," *Computers & Chemical Engineering*, vol. 27, no. 2, pp. 157–166, 2003.

[18] M. Diehl, H. J. Ferreau, and N. Haverbeke, "Efficient numerical methods for nonlinear MPC and moving horizon estimation," in *Nonlinear model predictive control*, 2009, pp. 391–417.

[19] M. L. Felis, "RBDL: an efficient rigid-body dynamics library using recursive algorithms," *Autonomous Robots*, vol. 41, no. 2, pp. 495–511, 2017.

[20] M. Felis, "Rigid Body Dynamics Library (RBDL)," 2012-2017. [Online]. Available: {https://bitbucket.org/MartinFelis/rbdl}

[21] D. B. Leineweber, I. Bauer, H. G. Bock, and J. P. Schlöder, "An efficient multiple shooting based reduced SQP strategy for large-scale dynamic process optimization. Part 1: theoretical aspects," *Computers & Chemical Engineering*, vol. 27, no. 2, pp. 157–166, 2003.

[22] D. Leineweber, A. Schäfer, H. Bock, and J. Schlöder, "An efficient multiple shooting based reduced SQP strategy for large-scale dynamic process optimization: Part II: Software aspects and applications," *Computers & chemical engineering*, vol. 27, no. 2, pp. 167–174, 2003.

[23] P. E. Gill, W. Murray, and M. A. Saunders, "User's guide for *QPOPT 1.0*: a FORTRAN package for quadratic programming," Department of Operations Research, Stanford University, Tech. Rep. SOL 95-4, 1995.

[24] W. Caarls, "Generic Reinforcement Learning library," 2015-2017. [Online]. Available: {https://github.com/wcaarls/grl}

[25] I. Koryakovskiy, M. Kudruss, R. Babuška, W. Caarls, C. Kirches, K. Mombaur, J. P. Schlder, and H. Vallery, "Benchmarking model-free and model-based optimal control," *Robotics and Autonomous Systems*, vol. 92, pp. 81–90, 2017.