Efficient Whole-Body Trajectory Optimization Using Contact Constraint Relaxation

Michael Neunert, Farbod Farshidian and Jonas Buchli

***Abstract*— In this work we present a Trajectory Optimization framework for whole-body motion planning for floating base robots. We demonstrate how the proposed approach can optimize whole body trajectories for most common gaits found in bipeds and quadrupeds. The underlying optimal control problem is solved efficiently using Sequential Linear Quadratic control. In contrast to most previous methods, the proposed approach is fast while still using a full dynamic model. Additionally, the approach is contact model free. Instead contact forces are added to the Optimal Control formulation as additional control inputs and contact constraints are handled using a relaxation approach. In our experiments we demonstrate that, despite this relaxation, our solver resolves constraint violations in only a few iterations. Hardware experiments underline the transferability from simulation to hardware.**

I. Introduction

Trajectory Optimization (TO) is a highly popular approach for solving complex, dynamic motion planning problems in robotics. It promises to replace manually designed planning and control frameworks. In TO non-linear optimization or Optimal Control are used to design a trajectory that minimizes a high level performance criterion, e.g. a cost function. Also in legged robotics, TO is gaining a lot of attention. However, due to the high number of degrees of freedom of modern legged robots and their need for establishing and breaking contacts with the environment, these problems become high-dimensional and non-convex. Thus, different approaches, both for the formulation of the problem as well as for solving it, have been proposed.

A. Related Work

Coros et al. [1] propose an optimization approach on a hierarchy of simplified models, including an inverted pendulum model, to solve whole-body motion planning tasks on a humanoid robot. Their approach shows impressive results and also optimizes contact locations and timings. However, it also relies on heuristics and optimality cannot be guaranteed. Another gait free approach that relies on an inverted pendulum model is presented by Mordatch et al. [2]. A different way of reducing the model complexity is using centroidal dynamics, e.g. as used in [3] and [4], leading to great hardware results. Another approach using a simplified model without pre-specifying contact sequences is proposed by Mordatch et al. in [5]. However, the proposed method additionally relaxes the entire optimization problem and only enforces dynamic and kinematic consistency as soft constraints, leading to severe violations of these quantities. A whole-body trajectory optimization approach is presented by Posa et al. [6]. In this work, the authors apply direct transcription and the resulting non-linear programming is solved using Sequential Quadratic Programming (SQP). While not assuming a given contact sequence, the approach is only demonstrated on lower dimensional, planar systems and might not scale to robots with many degrees of freedom. Other direct transcription approaches using SQP are presented in [7] and [8]. Both approaches follow the idea of projecting their decision variables to the tangent of the constraint manifold. In both cases, results are presented for full models of complex robots, i.e. a humanoid and a quadruped respectively. Therefore, solving the resulting non-linear program is computationally very costly. The same drawback also applies to the Multiple shooting approach in [9] applied to the humanoid robot HRP-2. An even more complex robot model that includes actuator dynamics is used by Gehring et al. [10]. Also here, the high model dimensionality leads to long computation times. Recently, researchers have tried to tackle Trajectory Optimization using Optimal Control algorithms. Popular for their efficiency are Differential Dynamic Programming (DDP) [11] and closely related algorithms such as iterative Linear Quadratic Gaussian (iLQG) [12] and Sequential Linear Quadratic (SLQ) [13] control. Impressive results computed close to real-time have been obtained from such solvers [14]. In recent work [15] we have demonstrated that such an approach can also automatically discover gaits and the resulting trajectories can be applied to hardware.

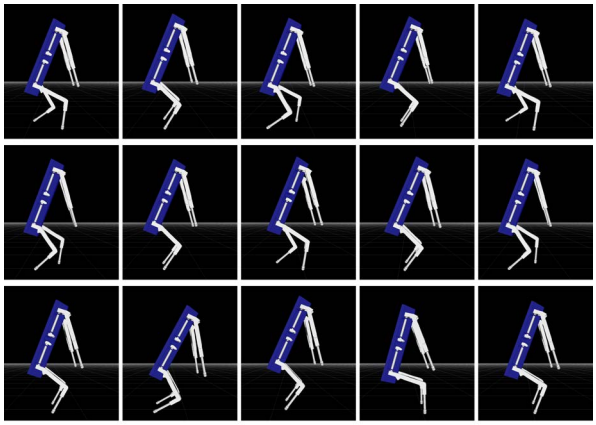

Fig. 1. Time series of the optimized trajectories for the humanoid gaits. The picture series progresses from left to right and the robot is moving to the right. From top to bottom the gaits are: walk, run and hop.

This research has been funded through a Swiss National Science Foundation Professorship award to Jonas Buchli, the NCCR Robotics, and a Max-Planck ETH Center for Learning Systems fellowship to Farbod Farshidian.

All authors are at the Agile & Dexterous Robotics Lab, ETH Zurich, Switzerland {neunertm, farbodf, buchlij} @ ethz.ch

However, in such methods there is a trade-off to be made between using a physically accurate, stiff contact model or softening it for better gradients and faster convergence. Additionally, these approaches require very careful cost function engineering to obtain reasonable solutions. To overcome the limitations of handling constraints in such methods, we have derived a continuous–time SLQ variant which allows for including state–input constraints as hard limits [16]. Furthermore, this work provides an algorithmic foundation for gradually enforcing state constraints. While we have applied this approach to a centroidal model of a quadruped in simulation, we have not yet tested it on a full multi–rigid body model or on hardware.

B. Contribution

In this work, we propose how to efficiently solve a TO problem over contact switches using SLQ. We present a contact constraint relaxation scheme by including contact forces as control input variables and gradually enforcing contact constraints via the cost function. While such approaches are said to have poor numeric properties and might lead to constraint violation, we demonstrate how our method reliably optimizes constraint satisfactory trajectories. We illustrate this for common gait types of bipeds and a quadrupeds. Due to the relaxation and an efficient Sequential Linear Quadratic control solver, the non-linear TO problem is solved in a few seconds for a full non-linear dynamic model. This outperforms many state-of-the-art TO approaches. By using a single quadratic cost across different gaits, we show that the proposed formulation does not require extensive tuning. While gait patterns need to be pre-specified, we still optimize for contact locations without relying on heuristics or model reductions. Lastly, this work is a rare example where optimized gaits obtained from whole-body TO are directly applied to hardware.

II. Trajectory Optimization

A. Optimal Control Problem

In our Trajectory Optimization problem, we assume general non-linear system dynamics

$$\boldsymbol{x}(t+1) = f(\boldsymbol{x}(t), \boldsymbol{u}(t)), \quad \boldsymbol{x}(0) = \boldsymbol{x}_0 \tag{1}$$

where $x(t)$ and $u(t)$ represent state and control input respectively. For this system, we are trying to find a time-varying feedforward and feedback controller of the following form

$$\boldsymbol{u}(\boldsymbol{x}, t) = \boldsymbol{u}_{ff}(t) + \mathcal{K}(t)\boldsymbol{x} \tag{2}$$

where $u_{ff}(t)$ denotes a time-varying feedforward action and $\mathcal{K}(t)$ a time-varying feedback controller. Given the system dynamics, we obtain such a controller by solving the finite time-horizon optimal control problem

$$J(\boldsymbol{x}, \boldsymbol{u}) = \Phi\left(\boldsymbol{x}(t_f)\right) + \sum_{t=0}^{t_f-1} L\left(\boldsymbol{x}(t), \boldsymbol{u}(t)\right) \tag{3}$$

where $\Phi(\cdot)$ describes the final evaluated at the end of the time horizon t_f and $L(\cdot)$ describes the intermediate cost.

Algorithm 1 SLQ Algorithm

Given
- System dynamics as in Eq. 1, Cost function as in Eq. 4
- Initial stable control law, $\boldsymbol{u}(\boldsymbol{x}, t)$

repeat
- Forward simulate the system dynamics
- Linearize the system dynamics along the trajectory:
  $\delta\boldsymbol{x}(t+1) = \boldsymbol{A}(t)\delta\boldsymbol{x}(t) + \boldsymbol{B}(t)\delta\boldsymbol{u}(t)$
- Quadratize cost function along the trajectory $\boldsymbol{\tau}$:
  $\tilde{J} \approx p(t) + \delta\boldsymbol{x}^\top(t_f)\boldsymbol{p}(t_f) + \frac{1}{2}\delta\boldsymbol{x}^\top(t_f)\boldsymbol{P}(t_f)\delta\boldsymbol{x}(t_f)$
  $+ \sum_{t=0}^{t_f-1} q(t) + \delta\boldsymbol{x}^\top\boldsymbol{q}(t) + \delta\boldsymbol{u}^\top\boldsymbol{r}(t)$
  $+ \frac{1}{2}\delta\boldsymbol{x}^\top\boldsymbol{Q}(t)\delta\boldsymbol{x} + \frac{1}{2}\delta\boldsymbol{u}^\top\boldsymbol{R}(t)\delta\boldsymbol{u}$
- Backwards solve the Riccati-like difference equations:
  $\boldsymbol{P}(t) = \boldsymbol{Q}(t) + \boldsymbol{A}^\top(t)\boldsymbol{P}(t+1)\boldsymbol{A}(t) +$
  $\boldsymbol{K}^\top(t)\boldsymbol{H}\boldsymbol{K}(t) + \boldsymbol{K}^\top(t)\boldsymbol{G} + \boldsymbol{G}^\top\boldsymbol{K}(t)$
  $\boldsymbol{p}(t) = \boldsymbol{q} + \boldsymbol{A}^\top(t)\boldsymbol{p}(t+1) + \boldsymbol{K}^\top(t)\boldsymbol{H}\boldsymbol{l}(t) + \boldsymbol{K}^\top(t)\boldsymbol{g} + \boldsymbol{G}^\top\boldsymbol{l}(t)$
  $\boldsymbol{H} = \boldsymbol{R}(t) + \boldsymbol{B}^\top(t)\boldsymbol{P}(t+1)\boldsymbol{B}(t)$
  $\boldsymbol{G} = \boldsymbol{B}^\top(t)\boldsymbol{P}(t+1)\boldsymbol{A}(t), \quad \boldsymbol{g} = \boldsymbol{r}(t) + \boldsymbol{B}^\top(t)\boldsymbol{p}(t+1)$
  $\boldsymbol{K}(t) = -\boldsymbol{H}^{-1}\boldsymbol{G}, \quad \boldsymbol{l}(t) = -\boldsymbol{H}^{-1}\boldsymbol{g}$
- Update the control:
  $\boldsymbol{u}(\boldsymbol{x}, t) = \boldsymbol{u}_n(t) + \alpha\boldsymbol{l}(t) + \boldsymbol{K}(t)\left(\boldsymbol{x}(t) - \boldsymbol{x}_n(t)\right)$
- Line search over α
- Increase constraint penalty: $\gamma \leftarrow \gamma_0\,\gamma$

until maximum number of iterations or converged ($\boldsymbol{l}(t) < l_t$)

B. Sequential Linear Quadratic Control

In this work, we are solving the non-linear Optimal Control problem stated above by applying Sequential Linear Quadratic (SLQ) control [13]. Given an initial guess, SLQ forward simulates the non-linear system dynamics. Afterwards, a linear approximation of the dynamics and a quadratic approximation of the cost function is computed around the obtained trajectory. This results in a series of Linear Quadratic control problems which are then solved with a backward sweep using Ricatti-like equations. This backwards sweep results in a feedforward control update and a new feedback controller. Solving Ricatti-like equations makes SLQ very efficient and results in linear complexity with respect to the time horizon. One drawback of SLQ is that it cannot handle hard state constraints. However, using a constraint relaxation described in Subsection II-C they can be included as soft constraints in the cost function. Our soft-constrained SLQ version is summarized in Algorithm 1. As we will show later, despite using such soft constraints, the algorithm still finds constraint satisfactory trajectories. While handling hard input constraints in SLQ-type algorithms is possible, as shown in [16] and [17], we apply constraint relaxation on input constraints as well to promote convergence speed.

C. Cost Function and Constraint Relaxation

In this work, we assume a quadratic cost function with an added constraint violation penalty

$$J = \bar{\boldsymbol{x}}(t_f)^\top\boldsymbol{Q}_f\bar{\boldsymbol{x}}(t_f) + \sum_{t=0}^{t_f-1}\boldsymbol{x}(t)^\top\boldsymbol{Q}\boldsymbol{x}(t) + \boldsymbol{u}(t)^\top\boldsymbol{R}\boldsymbol{u}(t) + \gamma_{(i)}\sum_{t=0}^{t_f} C_c(\boldsymbol{x}, \boldsymbol{u}, t), \tag{4}$$

where $\bar{\boldsymbol{x}}(t_f) = \boldsymbol{x}_f(t_f) - \boldsymbol{x}_{f,d}$ is the state deviation from desired final state. $\boldsymbol{Q}_f$, $\boldsymbol{Q}$ and $\boldsymbol{R}$ denote weighting matrices for final cost, intermediate cost and input cost respectively. The quadratic elements of the cost function encode the task of the robot and are able to regularize states if required. Additionally, we add a constraint violation cost term $C_c(\boldsymbol{x}, \boldsymbol{u}, t)$

which allows for handling both state and input constraints using relaxation. The constraints considered in this work and the resulting soft constraint function are described in Subsection III-C. In early iterations of the algorithm, constraints are relaxed to allow the solver to find a good initial guess that fulfills the task but may violate constraints. Over the course of the iterations, the constraints are then enforced by increasing the constraint violation cost gain $\gamma_{(i)}$ after each iteration. We increase the gain exponentially: $\gamma_{(i)} = \gamma_0^i$, where i denotes the iteration number and γ_0 a user defined value. In our examples, we choose $\gamma_0 = 3$ as an initial gain.

III. System Model and Contact Constraints

To verify our approach, we use our hydraulically actuated, quadrupedal robot HyQ [18], modelled using rigid body dynamics. This four legged robot has three actuated joints per leg which are all torque controlled.

A. Rigid Body Dynamics

Due to a high performance onboard joint torque controller, we assume HyQ to be a rigid body system with perfect torque inputs, described as

$$\dot{\boldsymbol{q}}(t+1) = \boldsymbol{M}^{-1}\left(\boldsymbol{J}_c^\top \boldsymbol{\lambda} + \boldsymbol{S}^\top \boldsymbol{\tau} - \boldsymbol{h}\right)\Delta t \tag{5}$$

where $\boldsymbol{M}$ is the joint space inertia matrix, $\boldsymbol{h}$ are the Coriolis, centrifugal and gravity terms and $\boldsymbol{q}$ is the state vector containing base and joint state. Contact forces $\boldsymbol{\lambda}$ act on the system via the Jacobian $\boldsymbol{J}_c$, whereas actuator forces $\boldsymbol{\tau}$ get mapped to the system by the input selection matrix $\boldsymbol{S}$.

In order to satisfy our formulation of a non-linear system in Equation 1, we define our state in the optimization as

$$\boldsymbol{x} = [{}_{\mathcal{W}}\boldsymbol{q}\ {}_{\mathcal{L}}\dot{\boldsymbol{q}}]^\top = [{}_{\mathcal{W}}\boldsymbol{q}_B\ {}_{\mathcal{W}}\boldsymbol{x}_B\ \boldsymbol{q}_J\ {}_{\mathcal{L}}\omega_B\ {}_{\mathcal{L}}\dot{\boldsymbol{x}}_B\ \dot{\boldsymbol{q}}_J]^\top \tag{6}$$

where ${}_{\mathcal{W}}\boldsymbol{q}_B$ and ${}_{\mathcal{W}}\boldsymbol{x}_B$ represent the pose of the trunk expressed in a global inertial "world" frame $\mathcal{W}$. We use Euler angles (roll-pitch-yaw) to represent the base orientation. ω_B and $\dot{\boldsymbol{x}}_B$ jointly represent the base twist, i.e. angular and linear velocity, expressed in a local body frame $\mathcal{L}$. $\boldsymbol{q}_J$ and $\dot{\boldsymbol{q}}_J$ denote joint angles and velocities respectively. Combining Equations 1, 5 and 6 our system dynamics can be expressed as

$$\boldsymbol{x}(t+1) = \boldsymbol{x}(t) + \begin{bmatrix} \boldsymbol{R}_{\mathcal{WL}}\ {}_{\mathcal{L}}\dot{\boldsymbol{q}}(t) \\ \boldsymbol{M}^{-1}(\boldsymbol{S}^\top\boldsymbol{\tau} + \boldsymbol{J}_c^\top\boldsymbol{\lambda} - \boldsymbol{h}) \end{bmatrix}\Delta t \tag{7}$$

where $\boldsymbol{R}_{\mathcal{WL}}$ rotates a vector expressed in the local body frame $\mathcal{L}$ to the inertial "world" frame $\mathcal{W}$.

B. Implicit Contact Model

If we were to use an explicit contact model, the contact force vector $\boldsymbol{\lambda}$ in Equations 5 and 7 would become a function of the rigid body position $\boldsymbol{q}$ and velocity $\dot{\boldsymbol{q}}$ vectors. The issue with such a contact model is that it needs to be stiff to reflect reality but still sufficiently soft to help convergence of any optimal control or optimization problem. In order to avoid this conflict, we do not define an explicit contact model. Instead, we add the contact force vector $\boldsymbol{\lambda}$ to our control input vector $\boldsymbol{u}(t) = [\boldsymbol{\tau}\ \boldsymbol{\lambda}]^\top$. Furthermore, the contact constraints (see Subsection III-C) are added to the problem to ensure physicality.

Using this implicit contact model has both benefits and drawbacks. One major benefit is that it allows the Optimal Control solver to directly influence the contact forces which relaxes the otherwise very stiff problem. This allows for longer integration steps and faster convergence rates. However, it also makes the Optimal Control problem higher dimensional. This issue is partially mitigated since we do not require to compute a contact model. State-of-the-art simulators often solve Linear Complementary Problems (LCP) in their contact models which require iterative solvers. Here, we combine the contact model solving and trajectory optimization in a single Optimal Control problem.

One might be inclined to use an explicit contact model and increase its stiffness over the course of iterations in order to get the best of both worlds. However, such an approach is difficult to apply to an SLQ-type approach. In each iteration, SLQ relies on a rollout of the dynamics given a controller obtained in the last iteration. If we increase the contact model stiffness between iterations, the rollout in the following iteration might be unstable and will not provide good linearization points for the control update.

C. Contact Constraints

We assume a rigid body system is in contact with its environment if the distance between the system and the environment is zero, i.e. there is no gap between both but also penetration is inadmissible. Only when in contact, external forces can act between the system and the environment. These two requirements describe a rigid contact model which can be expressed by a single complementary constraint

$$d_{ee}(\boldsymbol{q})\boldsymbol{\lambda}_{ee} = 0 \tag{8}$$

where $\boldsymbol{\lambda}_{ee}$ is the contact force acting at a specific end-effector and $d_{ee}(\boldsymbol{q})$ is the distance of the end-effector to the environment as a function of the rigid body state vector $\boldsymbol{q}$ introduced in Equation 5. While this is a very general formulation, it can make the TO problem highly non-convex. To mitigate this issue, we assume a given contact sequence, i.e. a binary contact state $c_{ee}(t)$ for each end-effector. This contact state function evaluates to 1 if an end-effector is in contact and 0 if it is not in contact. This allows us to split the complementary constraint into its two original parts, a distance constraint and a force constraint

$$c_{ee}(t)d_{ee}(\boldsymbol{q}) = 0, \quad (1 - c_{ee}(t))\boldsymbol{\lambda}_{ee} = 0. \tag{9}$$

Furthermore, under the assumption that no slippage occurs, the point $\boldsymbol{p}_{ee}(\boldsymbol{q}, \dot{\boldsymbol{q}})$ on a rigid body system which is in contact with the environment is constrained to have zero velocity

$$c_{ee}(t)\dot{\boldsymbol{p}}_{ee}(\boldsymbol{q}, \dot{\boldsymbol{q}}) = 0. \tag{10}$$

This constraint separation converts the state-input complementary constraint (Eq. 8) into pure state and input constraints (Eq 9). Given our constraint relaxation formulation introduced in Subsection II-C, we define the constraint

Fig. 2. Time series of the optimized trajectories for the quadruped gaits. The picture series progresses from left to right and the robot moves to the right in the image plane. From top to bottom the gaits are: trot (hardware), flying trot (simulation), hop, bound, rotary gallop and transverse gallop.

violation term of our cost function (Eq. 4) as the sum of squared violations over all end-effectors

$$\boldsymbol{C}_c(\boldsymbol{x},\boldsymbol{u},t) = \sum_{ee=0}^{N_{ee}} (1-c_{ee}(t))(\beta_3 \boldsymbol{\lambda}_{ee}^\top \boldsymbol{\lambda}_{ee} + \beta_4 d_{s,ee}^2(t)) + c_{ee}(t)(\beta_1 \dot{\boldsymbol{p}}_{ee}^\top \dot{\boldsymbol{p}}_{ee} + \beta_2 d_{ee}^2) \quad (11)$$

where N_{ee} is the number of end-effectors and $\beta_{1\ldots4}$ are constant weighting factors. Additionally to the constraint, we add a desired swing leg height $d_{s,ee}$ to the cost function.

IV. Experiments

To evaluate the proposed approach, we optimize trajectories for the common dynamic gait patterns found in bipedal and quadrupedal locomotion. In both cases, we use the model of our robot HyQ described in Section III-A. While by design, HyQ is a quadruped, we use the same model for optimizing humanoid gaits. While in reality, the robot is not strong enough for bipedal locomotion, it allows us to compare bipedal and quadrupedal locomotion. For each of the two systems, we only use one single cost function with fixed, purely diagonal weighting matrices to optimize all gaits. Furthermore, we demonstrate the transferability of the optimized motion on several trotting tasks executed on hardware. The cost function weights used for hardware experiments are the same as in simulation. To set the goal for the robot to reach, we specify the desired final body pose in the cost. We cold start all tasks with zero control input resulting in a vertical fall. The SLQ and integration step size is et to $\Delta t = 2ms$. The obtained results are summarized below and shown in the video attachment (`https://youtu.be/FbC8FZnkGks`).

A. Humanoid

First, we demonstrate humanoid gaits in simulation. We rely on the three most common gaits also found in humans, namely walking, running and hopping. Picture series of all three optimized gaits are shown in Figure 1.

1) Walk: First we show a humanoid walking gait with 50% contact duty cycle and without flight or double support phase. While the contact sequence is pre-specified, the contact locations are not. This allows the optimizer to find a trajectory where the robot first steps back with its left leg in order to create forward thrust during the stance phase. The subsequent footstep locations are fairly equidistant until the robot returns to its initial configuration at the end of the trajectory. Since our humanoid model does not have ankles to produce torques, we can observe rhythmic, lateral tilting of the base. Since HyQ's "arms" (front legs) are lightweight and we penalize motions of the corresponding joints, the optimization keeps them close to their initial configuration during the motion. If more arm motion is desired, the cost function weights can be adjusted accordingly.

2) Run: For the humanoid running gait, we reduce the contact duty cycle of the walk to 40%, leading to two flight phases of 10%, resulting in a total flight time of 20% of the cycle. As a result, the gait is more dynamic and the robot shows a small jumping motion between support phases. We still observe the left leg stepping back to initialize the gait.

3) Hop: The last gait sequence we demonstrate on a humanoid is a symmetric hopping gait. Here, the contact duty cyle is set to 50% for both legs. In the optimized trajectory, the robot first hops back again to generate forward thrust. Afterwards, it executes two longer hops and then a shorter one to return to its resting posture at the goal position.

B. Quadruped

For the quadruped case, we demonstrate variations of four basic gait pattern types, resulting in six examples in simulation in total. Figure 2 shows a picture series for five gaits: trot, hop, bound, rotary gallop and transverse gallop. While the galloping gait exceeds the hardware limitations, we perform several hardware tests of the trotting gait.

1) Trot and Running Trot: As a first example on the quadruped, we show a trot where diagonal legs are moved simultaneously. As with the humanoid walk we first demonstrate a gait pattern with no overlap, i.e. without flight phases or full support phases. The resulting trajectory is a trot that starts and ends in a four leg stance. From Figure 3 we can see that control effort is mainly required during the stance phase. The strongest torques can be seen in the Knee Flexion-Extension joint. Figure 5 shows the optimized contact forces. The largest contact forces occur during touchdown and just before lift-off. This is directly correlated with the joint torques in Figure 3. As expected, the contact forces in z-direction dominate, which corresponds to the vertical axis in world coordinates. However, most importantly, the contact force plots clearly demonstrate that forces are only applied during contact phases. This underlines that the contact relaxation method is successful in avoiding contact constraint violations. We also demonstrate a trot in simulation where we set the goal position to the front-left of the initial position and set the desired final yaw angle to 90 degrees. However, the cost function weights remains the same. As a result, HyQ trots in a curved motion. By reducing the contact duty cycle of the trot to 35%, we can also obtain a running trot with two flight phases.

To show that the optimized trajectories are physical and are directly applicable to the robot, we perform trotting tests

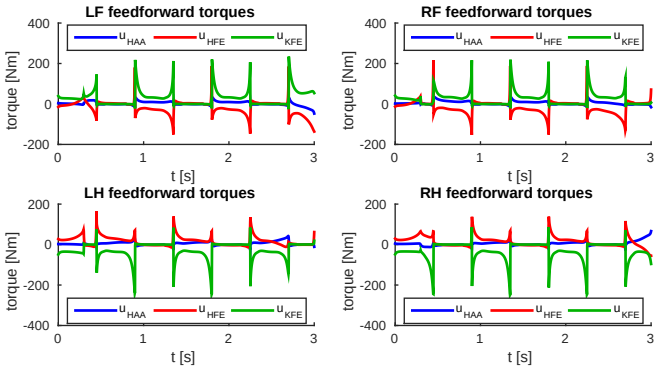

Fig. 3. Feedforward joint torques Hip-Abduction-Adduction (HAA), Hip-Flexion-Extension (HFE) and Knee-Flexion-Extension (KFE) for all four legs of a quadruped during a trot. During the trot, the highest joint torques can be observed in the KFE joint.

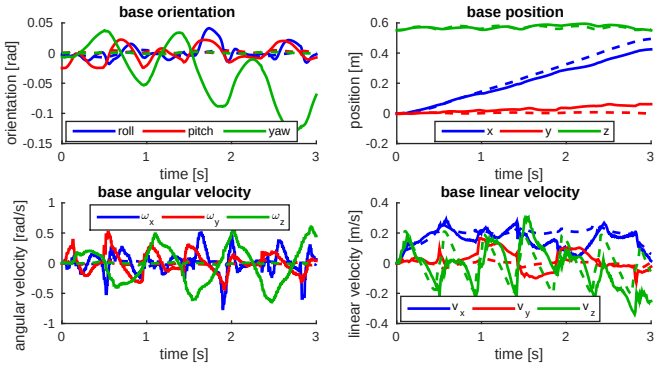

Fig. 4. Comparison between planned (dashed) and measured (solid) base state trajectories for a trot executed on hardware. Using the full body model for optimizing the trajectories allows for executing them without base state feedback for several seconds.

on hardware. We test two variations of the trot: One with 50% duty cycle, i.e. no overlap of stance or flight phases and one with 60% duty cycle, i.e. a very brief four leg support phase. In both cases, we directly apply the optimized input trajectory as a feedforward torque command to the actuator. Additionally, we add a joint space PD controller to track joint positions. For reasons discussed in [15], we are not using the feedback control gains provided by SLQ. Figure 4 compares planned and measured base trajectories. While a base controller could enhance performance, our full model based approach allows us for successfully executing the trajectories for several seconds without base feedback.

2) Hop: As a next task, we demonstrate symmetric hopping where the contact sequence is the same for all four legs. The resulting trajectory shows an almost constant forward velocity. This greatly reduces the involved control effort. If we decrease control effort weightings, we observe that the base almost comes to a complete stop during stance phases.

3) Bound: Another quadruped gait is bounding. Here the legs move in pairs, grouped in front and hind legs. Between two subsequent stance phases, there is a short flight phase. We observe a rhythmic pitching motion in the base. Interestingly, the optimized trajectory shows almost equidistant jumps except for the last jump, which is executed in place at the goal position. A possible reason could be that

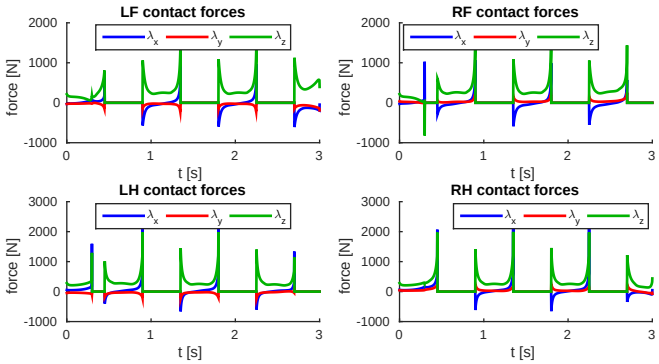

Fig. 5. Contact forces during a trot of a quadrupedal robot. Corresponding to Figure 3 contact forces are high when joint torques are high. The z-direction corresponding to the vertical axis in world coordinates, is dominating. No contact forces are exerted on legs in swing phase.

it is favorable to move with higher forward velocity given the dynamics and our cost function weightings. Reducing the time horizon by one gait cycle could potentially get rid of the final extra jump.

4) Rotary and Transverse Gallop: As a final test, we demonstrate rotary and transverse galloping. Both tasks differ only in the gait pattern of the left front leg. These gaits allow the robot to travel a large distance using long stride lengths. When compared to a trot, these long stride lengths result in relatively slow joint velocities given the speed of the base. Comparing the results with galloping animals, we see significant similarities. One similarity is that the base is pitched downwards around the time both front legs are in stance and pitched upwards when the hind legs are in stance.

C. Constraint Satisfaction

Since we are using soft constraints, we have to ensure that the obtained motions do not violate these constraints and result in unphysical trajectories. In the first iteration, we do not enable our soft constraints but allow the algorithm to find a good initial guess. Hence, after the first iteration the robot to "flies" to the goal position. For the second iteration we then enable the soft constraints. In the subsequent iterations, the total costs of the found solution stay flat or slightly increase since the algorithm is primarily trying to fix the constraint violation. So instead of looking at the cost function decay, we analyze the constraint violation cost. We focus on the quadruped results, since they involves more constraints to be satisfied. The results for the humanoid model are comparable. Figure 6 illustrate the constraint violations in input and end-effector position/velocity for all quadruped tasks as the Mean Squared Error (MSE). For the kinematic error, this is the sum of both the distance and velocity violation accumulated over all four legs. Similarly, the contact force constraint violation is also summed up over all four legs. As shown in the graphs, the algorithm finds solutions with constraint violations below levels of 10^{-3} in less than 18 iterations. To put these numbers in perspective, an unmodelled trunk weight of 0.01 kg would create an MSE in contact forces of 0.01. Kinematic violations are below the system's control accuracy. Also hardware tests show that constraint violations are negligible.

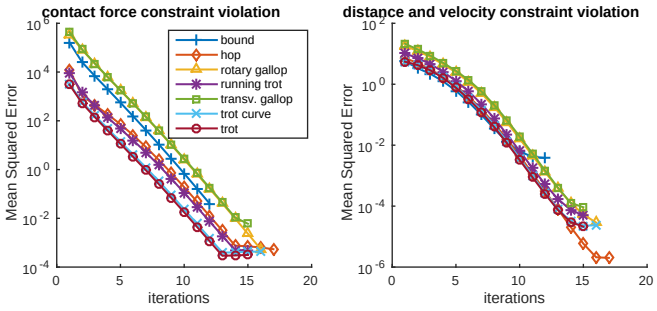

Fig. 6. MSE of the constraint violation for the quadruped gaits. The left plot shows the sum of contact force constraint violations. The right plot shows the sum of the end-effector constraint violations. In both cases the violation monotonically decreases to usually below levels of 10^{-3}.

D. Runtime

Theory suggests that the runtime of SLQ scales linear with the time horizon. In previous work [15] we have demonstrated that this also holds true for our SLQ implementation. However, the runtime-per-iteration differs. On one hand, our control input dimensionality is increased compared to the previous approach and our cost function is more complex. On the other hand, we can take larger integration steps and we do not need to compute a contact model. In order to check whether the approach is fast enough to do online planning or even re-optimizing the trajectory during execution in Model Predictive Control (MPC) fashion, we measure the runtime of each iteration. The profiling of the algorithm is done for the quadruped case since it is the higher dimensional model. Given a time horizon of 3 seconds and a sampling frequency of 500 Hz, our solver requires around 800 ms per iteration on an Intel Core i7-4810MQ processor. As expected, this is slightly worse than the runtime presented in our previous work. However, given that we can take larger integration steps, the presented approach outperforms our previous one in runtime-per-iteration given the same time horizon.

As shown in the previous subsection, the presented examples converge to negligible constraint violation in 10 to 18 iterations from cold start. Thus, the overall computation time until convergence usually lies in the range of 8 to 15 seconds for trajectories of 3 seconds. In our examples, we cold start, use small integration steps and long time horizons. Considering that in an MPC approach we would use shorter time horizons and a very good initial guess is available from the previous iteration, we could possibly run the approach in MPC fashion with the current implementation. Additionally, part of the system dynamics linearization is using numerical differentiation which could be replaced by analytical derivatives. Also, hardware results show that the trajectories are robust, permitting low MPC update rates.

V. Discussion, Conclusion and Outlook

In this work, we have presented how an efficient Trajectory Optimization algorithm can be combined with constraint relaxation to solve complex motion planning tasks under switching contacts in a few seconds. Despite this relaxation, constraint violations are robustly reduced to a negligible amount. Since we are not using an explicit contact model, the problem appears to be more convex resulting in fast convergence. Compared to our previous work [15], the problem is also less sensitive to cost function parameters. However, this comes at the expense of having to pre-specify the contact sequence which can be both an advantage or a disadvantage. Hardware results demonstrate that the trajectories can be directly applied to the physical system. So far we are not considering contact friction, joint torque or kinematic limits. These could also be included as relaxed constraints or as shown in [16]. By including pre-specified contact locations as constraints, we are planning to apply this approach to rough terrain locomotion, possibly even in MPC fashion.

References

[1] D. Zimmermann, S. Coros, Y. Ye, R. W. Sumner, and M. Gross, "Hierarchical planning and control for complex motor tasks," in *Proceedings of ACM SIGGRAPH*, 2015.

[2] I. Mordatch, M. de Lasa, and A. Hertzmann, "Robust physics-based locomotion using low-dimensional planning," in *Proceedings of ACM SIGGRAPH*, 2010.

[3] H. Dai, A. Valenzuela, and R. Tedrake, "Whole-body motion planning with centroidal dynamics and full kinematics," in *IEEE-RAS International Conference on Humanoid Robots*, 2014.

[4] T. Koolen, J. Smith, G. Thomas, S. Bertrand, J. Carff, N. Mertins, D. Stephen, P. Abeles, J. Englsberger, S. Mccrory *et al.*, "Summary of team IHMC's virtual robotics challenge entry," in *13th IEEE-RAS International Conference on Humanoid Robots (Humanoids)*, 2013.

[5] I. Mordatch, E. Todorov, and Z. Popović, "Discovery of complex behaviors through contact-invariant optimization," *ACM Transactions on Graphics*, 2012.

[6] M. Posa, C. Cantu, and R. Tedrake, "A direct method for trajectory optimization of rigid bodies through contact," *The International Journal of Robotics Research*, 2014.

[7] M. Posa, S. Kuindersma, and R. Tedrake, "Optimization and stabilization of trajectories for constrained dynamical systems," in *IEEE International Conference on Robotics and Automation (ICRA)*, 2016.

[8] D. Pardo, M. Neunert, A. W. Winkler, and J. Buchli, "Projection based whole body motion planning for legged robots," *arXiv preprint arXiv:1510.01625*, 2015.

[9] K. H. Koch, K. Mombaur, and P. Soueres, "Optimization-based walking generation for humanoid robot," *IFAC Proceedings*, 2012.

[10] C. Gehring, S. Coros, M. Hutter, C. D. Bellicoso, H. Heijnen, R. Diethelm, M. Bloesch, P. Fankhauser, J. Hwangbo, M. A. Hoepflinger, and R. Y. Siegwart, "Practice Makes Perfect: An Optimization-Based Approach to Controlling Agile Motions for a Quadruped Robot," *IEEE Robotics & Automation Magazine*, 2016.

[11] D. Mayne, "A second-order gradient method for determining optimal trajectories of non-linear discrete-time systems," *International Journal of Control*, 1966.

[12] E. Todorov and W. Li, "A generalized iterative lqg method for locally-optimal feedback control of constrained nonlinear stochastic systems," in *Proceedings of the American Control Conference*. IEEE, 2005.

[13] A. Sideris and J. E. Bobrow, "An efficient sequential linear quadratic algorithm for solving nonlinear optimal control problems," *IEEE Transactions on Automatic Control*, 2005.

[14] Y. Tassa, T. Erez, and E. Todorov, "Synthesis and stabilization of complex behaviors through online trajectory optimization," in *IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2012.

[15] M. Neunert, F. Farshidian, A. W. Winkler, and J. Buchli, "Trajectory optimization through contacts and automatic gait discovery for quadrupeds," *arXiv preprint arXiv:1607.04537*, 2016.

[16] F. Farshidian, M. Neunert, A. W. Winkler, G. Rey, and J. Buchli, "An efficient optimal planning and control framework for quadrupedal locomotion," *arXiv preprint arXiv:1609.09861*, 2016.

[17] Y. Tassa, N. Mansard, and E. Todorov, "Control-limited differential dynamic programming," in *IEEE International Conference on Robotics and Automation (ICRA)*, 2014.

[18] C. Semini, N. G. Tsagarakis, E. Guglielmino, M. Focchi, F. Cannella, and D. G. Caldwell, "Design of HyQ –a hydraulically and electrically actuated quadruped robot," *Proceedings of the Institution of Mechanical Engineers, Journal of Systems and Control Engineering*, 2011.