Dynamic Locomotion in the MIT Cheetah 3 Through Convex Model-Predictive Control

Jared Di Carlo[1], Patrick M. Wensing[2], Benjamin Katz[3], Gerardo Bledt[1,3], and Sangbae Kim[3]

***Abstract*— This paper presents an implementation of model predictive control (MPC) to determine ground reaction forces for a torque-controlled quadruped robot. The robot dynamics are simplified to formulate the problem as convex optimization while still capturing the full 3D nature of the system. With the simplified model, ground reaction force planning problems are formulated for prediction horizons of up to 0.5 seconds, and are solved to optimality in under 1 ms at a rate of 20-30 Hz. Despite using a simplified model, the robot is capable of robust locomotion at a variety of speeds. Experimental results demonstrate control of gaits including stand, trot, flying-trot, pronk, bound, pace, a 3-legged gait, and a full 3D gallop. The robot achieved forward speeds of up to 3 m/s, lateral speeds up to 1 m/s, and angular speeds up to 180 deg/sec. Our approach is general enough to perform all these behaviors with the same set of gains and weights.**

I. Introduction

Control of highly dynamic legged robots is a challenging problem due to the underactuation of the body during many gaits and due to constraints placed on ground reaction forces. As an example, during dynamic gaits[4] such as bounding or galloping, the body of the robot is always underactuated. Additionally, ground reaction forces must always remain in a friction cone to avoid slipping. Current solutions for highly dynamic locomotion include heuristic controllers for hopping and bounding [1], which are effective, but difficult to tune; two-dimensional planar simplifications [2], which are only applicable for gaits without lateral or roll dynamics; and evolutionary optimization for galloping [3], which cannot currently be solved fast enough for online use. Recent results on hardware include execution of bounding limit cycles discovered offline with HyQ [4] and learned pronking, trotting, and bounding gaits on StarlETH [5].

Predictive control can stabilize these dynamic gaits by anticipating periods of flight or underactuation, but is difficult to solve due to the nonlinear dynamics of legged robots and the large number of states and control inputs. Nonlinear optimization has been shown to be effective for predictive control of hopping robots [6], humanoids [7], [8], and quadrupeds [9], with [9] demonstrating the utility of heuristics to regularize the optimization. Another common approach is to use both a high-level planner, such as in [10], [11] and a lower level controller to track the plan. More recently, the experimental results in [12] show that whole-body nonlinear MPC can be used to stabilize trotting and jumping.

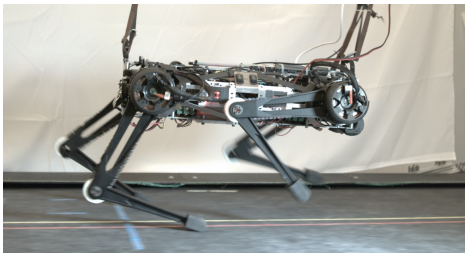

Fig. 1. The MIT Cheetah 3 Robot galloping at 2.5 m/s

The stabilization of the quadruped robot HyQ using convex optimization discussed in [13] demonstrates the utility of convex optimization, but the approach cannot be immediately extended to dynamic gaits due to the quasi-static simplifications made to the robot model. Similarly, in bipedal locomotion, convex optimization has been used to find the best forces to satisfy instantaneous dynamics requirements [14] and to plan footsteps with the linear inverted pendulum model [15] but the latter approach does not include orientation in the predictive model.

While galloping is well studied in the field of biology [16], [17], surprisingly few hardware implementations of galloping exist. The first robot to demonstrate galloping was the underactuated quadruped robot Scout II [18], which reached 1.3 m/s, but had limited control of yaw. The MIT Cheetah 1 robot [19] achieved high-speed galloping, but was constrained to a plane. To the best of our knowledge, the only previous implementation of a fully 3D gallop with yaw control is on the hydraulically actuated WildCat robot [20], developed by Boston Dynamics. Unfortunately, no specific details about WildCat or its control system have been published.

The main contribution of this paper is a predictive controller which stabilizes a large number of gaits, including those with complex orientation dynamics. On hardware, we achieved a maximum yaw rate of 180 deg/sec and a maximum linear velocity of 3.0 m/s during a fully 3D gallop, which we believe to be the fastest gallop of an electrically actuated robot, and the fastest angular velocity of any legged robot similar in scale to Cheetah 3. Our controller can be formulated as a single convex optimization problem which considers a 3D, 12 DoF model of the robot. The solution of the optimization problem can be directly used to compute

Authors are with the [1]Department of Electrical Engineering and Computer Science at the Massachusetts Institute of Technology, Cambridge, MA, 02139, USA; the [2]Department of Aerospace and Mechanical Engineering at the University of Notre Dame, Notre Dame, IN, 46556; and the [3]Department of Mechanical Engineering at the Massachusetts Institute of Technology, Cambridge, MA, 02139, USA. email: dicarloj@mit.edu, pwensing@nd.edu, sangbae@mit.edu

[4]In this paper, the term *dynamic gaits* is used to refer to gaits with significant periods of flight or underactuation.

This work was supported by National Science Foundation [NSF-IIS-1350879]

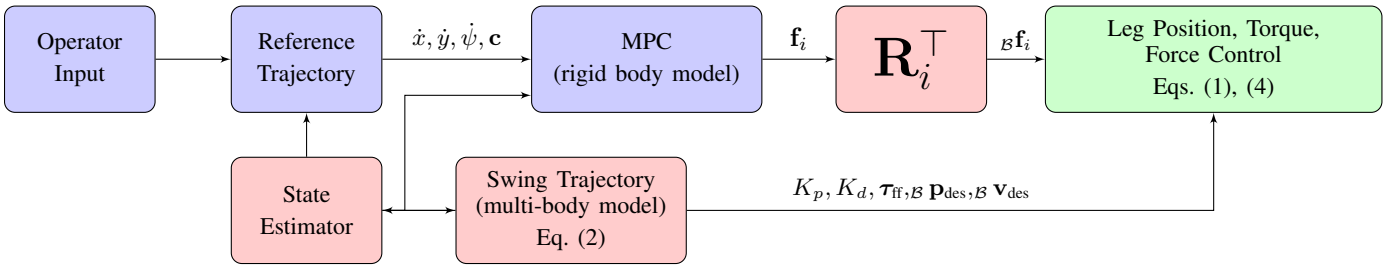

Fig. 2. Control system block diagram. The reference trajectory contains desired velocities and contact sequence from the operator. Blocks shaded blue run at 30 Hz, blocks shaded red run at 1 kHz, and blocks shaded green run at 4.5 kHz.

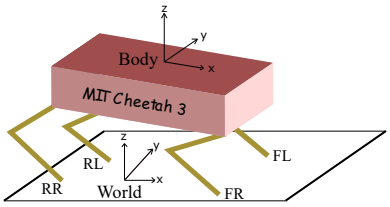

Fig. 3. World and Body Coordinate Systems

motor torques which stabilizes the robot without the use of an additional feedback controller.

The convex optimization approach has the benefit of avoiding nonlinear optimization techniques, which, unlike convex optimization solvers, are not guaranteed to find the global optimum and may suffer from numerical issues. Recent progress on convex optimization [21] and applications to model predictive control [22] has led to the development of many open source solvers such as ECOS [23] and qpOASES [24] that enable model predictive control problems to be solved rapidly and reliably.

The proposed controller was tested on the MIT Cheetah 3 quadruped robot on a treadmill and untethered on the ground, where it was able to stand, trot, flying-trot, pace, bound, pronk, and gallop. Aggressive turning maneuvers and omnidirectional locomotion were tested as well. The robot is powered by onboard batteries and all control calculations are done on an onboard processor. During tests, the robot encountered a number of obstacles and disturbances, including a staircase with full size steps covered in debris, which it was able to blindly climb with while doing a flying trot. To handle the large terrain disturbances, the robot used a contact detection algorithm described in [25]. For all tests, including stair climbing, the robot uses no sensors other than an inertial measurement unit (IMU), joint position encoders, and current sensors in the motor controllers, with the latter used only for a current control feedback loop. The supplemental video includes highlights from testing. Footage of complete tests is available online at: `https://www.youtube.com/watch?v=q6zxCvCxhic`.

This paper is organized as follows. Section II describes the control framework used on the Cheetah 3 robot. The dynamics model is presented in Section III, followed by the model predictive control problem formulation in Section IV. Section V contains implementation details and results.

II. Control Architecture

A. MIT Cheetah 3 Robot

The MIT Cheetah 3, shown in figure 1, is a 45 kg quadruped robot built by the MIT Biomimetic Robotics Lab. The robot utilizes the proprioceptive actuator technology developed for the Cheetah 2 robot [26], which had low inertia, easily backdrivable hip and knee actuators to mitigate impacts. Cheetah 3 uses the same actuator technology in its hips, knees, and additionally the ab/ad joints, which allows the robot to execute dynamic turns.

Each of the robot's four legs has three torque controlled joints: ab/ad, hip, and knee. Each joint can produce a maximum torque of 250 Nm and can operate at a maximum velocity of 21 rad/sec. Motor torque control is accomplished by adjusting the setpoint of the motor current control loop; there are no torque or force sensors.

B. Conventions

The body and world coordinate systems are defined in figure 3. All quantities in the body coordinate system have a left subscript $\mathcal{B}$. Quantities with no subscript are in the world coordinate system. Vectors are bold, upright, and lowercase $(\mathbf{a}, \boldsymbol{\omega})$, matrices are bold, upright, and uppercase $(\mathbf{A}, \boldsymbol{\Omega})$, and scalars are lowercase and italicized (a, ω). $\mathbf{1}_n$ is used to denote an $n \times n$ identity matrix.

C. State Machine

The controller proposed in this paper determines desired 3D ground reaction forces for behaviors with fixed-timing foot placement and liftoff. When a foot is scheduled to be in the air, the robot runs a swing leg controller, which is robust to early touchdowns. Otherwise, a ground force controller runs. These controllers are described in the following subsections and in Figure 2.

D. Swing Leg Control

The swing controller computes and follows a trajectory for the foot in the world coordinate system. The controller for tracking the trajectory uses the sum of a feedback and feedforward term to compute joint torques.

The control law used to compute joint torques for leg i is

$$\boldsymbol{\tau}_i = \mathbf{J}_i^\top \left[\mathbf{K}_p({}_\mathcal{B}\mathbf{p}_{i,\text{ref}} - {}_\mathcal{B}\mathbf{p}_i) + \mathbf{K}_d({}_\mathcal{B}\mathbf{v}_{i,\text{ref}} - {}_\mathcal{B}\mathbf{v}_i) \right] + \boldsymbol{\tau}_{i,\text{ff}} \tag{1}$$

where $\mathbf{J}_i \in \mathbb{R}^{3\times3}$ is the foot Jacobian, $\mathbf{K}_p, \mathbf{K}_d \in \mathbb{R}^{3\times3}$ are diagonal positive definite proportional and derivative gain matrices, ${}_{\mathcal{B}}\mathbf{p}_i, {}_{\mathcal{B}}\mathbf{v}_i \in \mathbb{R}^3$ are the position and velocity of the i-th leg, ${}_{\mathcal{B}}\mathbf{p}_{i,\text{ref}}, {}_{\mathcal{B}}\mathbf{v}_{i,\text{ref}} \in \mathbb{R}^3$ are the corresponding references for the position and velocity of the swing leg trajectory and $\boldsymbol{\tau}_{i,\text{ff}} \in \mathbb{R}^3$ is a feedforward torque, found with

$$\boldsymbol{\tau}_{i,\text{ff}} = \mathbf{J}^\top \boldsymbol{\Lambda}_i({}_{\mathcal{B}}\mathbf{a}_{i,\text{ref}} - \dot{\mathbf{J}}_i\dot{\mathbf{q}}_i) + \mathbf{C}_i\dot{\mathbf{q}}_i + \mathbf{G}_i \tag{2}$$

where $\boldsymbol{\Lambda}_i \in \mathbb{R}^{3\times3}$ is the operational space inertia matrix, $\mathbf{a}_{i,\text{ref}} \in \mathbb{R}^3$ is the reference acceleration in the body frame, $\mathbf{q}_i \in \mathbb{R}^3$ is the vector of joint positions, and $\mathbf{C}_i\dot{\mathbf{q}}_i + \mathbf{G}_i$ is the torque due to gravity and Coriolis forces for the leg.

To ensure high-gain stability in a wide range of leg configurations, the $\mathbf{K}_p$ matrix must be adjusted to keep the natural frequency of the closed loop system relatively constant in response to changes in the apparent mass of the leg. The i-th diagonal entry of $\mathbf{K}_p$ required to maintain a constant natural frequency ω_i for the i-th axis can be approximated with

$$K_{p,i} = \omega_i^2 \Lambda_{i,i} \tag{3}$$

where $\Lambda_{i,i}$ is the i,i-th entry in the mass matrix, corresponding to the apparent mass of the leg along the i-th axis.

Desired footstep locations are chosen with a simple heuristic described in Section V. Additionally, a contact detection algorithm [25] detects and handles early or late contact. If an early contact is detected, the controller immediately switches to stance and begins ground force control.

E. Ground Force Control

During ground force control, joint torques are computed with

$$\boldsymbol{\tau}_i = \mathbf{J}_i^\top \mathbf{R}_i^\top \mathbf{f}_i \tag{4}$$

where $\mathbf{R}$ is the rotation matrix which transforms from body to world coordinates, $\mathbf{J} \in \mathbb{R}^{3\times3}$ is the foot Jacobian, and $\mathbf{f}$ is the vector of forces calculated from the predictive controller in the world coordinate frame.

III. Simplified Robot Dynamics

The predictive controller models the robot as a single rigid body subject to forces at the contact patches. Although ignoring leg dynamics is a major simplification, the controller is still able to stabilize a high-DoF system and is robust to these multi-body effects. For the Cheetah 3 robot, this simplification is reasonable; the mass of the legs is roughly 10% of the robot's total mass. For each ground reaction force $\mathbf{f}_i \in \mathbb{R}^3$, the vector from the center of mass (COM) to the point where the force is applied is $\mathbf{r}_i \in \mathbb{R}^3$. The rigid body dynamics in world coordinates are given by

$$\ddot{\mathbf{p}} = \frac{\sum_{i=1}^n \mathbf{f}_i}{m} - \mathbf{g} \tag{5}$$

$$\frac{\mathrm{d}}{\mathrm{d}t}(\mathbf{I}\boldsymbol{\omega}) = \sum_{i=1}^n \mathbf{r}_i \times \mathbf{f}_i \tag{6}$$

$$\dot{\mathbf{R}} = [\boldsymbol{\omega}]_\times \mathbf{R} \tag{7}$$

where $\mathbf{p} \in \mathbb{R}^3$ is the robot's position, m is the robot's mass, $\mathbf{g} \in \mathbb{R}^3$ is the acceleration of gravity, $\mathbf{I} \in \mathbb{R}^3$ is the robot's inertia tensor, $\boldsymbol{\omega} \in \mathbb{R}^3$ is the robot's angular velocity, and $\mathbf{R}$ is the rotation matrix which transforms from body to world coordinates. $[\mathbf{x}]_\times \in \mathbb{R}^{3\times3}$ is defined as the skew-symmetric matrix such that $[\mathbf{x}]_\times\mathbf{y} = \mathbf{x}\times\mathbf{y}$ for all $\mathbf{x},\mathbf{y} \in \mathbb{R}^3$. The nonlinear dynamics in (6) and (7) motivate the approximations made in the following subsection to avoid the nonconvex optimization that would be required for nonlinear dynamics.

A. Approximated Angular Velocity Dynamics

The robot's orientation is expressed as a vector of Z-Y-X Euler angles [27] $\boldsymbol{\Theta} = [\phi\ \theta\ \psi]^\intercal$ where ψ is the yaw, θ is the pitch, and ϕ is the roll. These angles corresponds to a sequence of rotations such that the transform from body to world coordinates can be expressed as

$$\mathbf{R} = \mathbf{R}_z(\psi)\mathbf{R}_y(\theta)\mathbf{R}_x(\phi) \tag{8}$$

where $\mathbf{R}_n(\alpha)$ represents a positive rotation of α about the n-axis. The angular velocity in world coordinates can be found from the rate of change of these angles with

$$\boldsymbol{\omega} = \begin{bmatrix} \cos(\theta)\cos(\psi) & -\sin(\psi) & 0 \\ \cos(\theta)\sin(\psi) & \cos(\psi) & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} \dot{\phi} \\ \dot{\theta} \\ \dot{\psi} \end{bmatrix} \tag{9}$$

If the robot is not pointed vertically ($\cos(\theta) \neq 0$), equation (9) can be inverted to find

$$\begin{bmatrix} \dot{\phi} \\ \dot{\theta} \\ \dot{\psi} \end{bmatrix} = \begin{bmatrix} \cos(\psi)/\cos(\theta) & \sin(\psi)/\cos(\theta) & 0 \\ -\sin(\psi) & \cos(\psi) & 0 \\ \cos(\psi)\tan(\theta) & \sin(\psi)\tan(\theta) & 1 \end{bmatrix} \boldsymbol{\omega} \tag{10}$$

For small values of roll and pitch (ϕ, θ), equation (10) can be approximated as

$$\begin{bmatrix} \dot{\phi} \\ \dot{\theta} \\ \dot{\psi} \end{bmatrix} \approx \begin{bmatrix} \cos(\psi) & \sin(\psi) & 0 \\ -\sin(\psi) & \cos(\psi) & 0 \\ 0 & 0 & 1 \end{bmatrix} \boldsymbol{\omega} \tag{11}$$

which is equivalent to

$$\begin{bmatrix} \dot{\phi} \\ \dot{\theta} \\ \dot{\psi} \end{bmatrix} \approx \mathbf{R}_z(\psi)\boldsymbol{\omega} \tag{12}$$

Note that the order in which the Euler angle rotations are defined is important; with an alternate sequence of rotations, the approximation will be inaccurate for reasonable robot orientations.

Equation (6) can be approximated with:

$$\frac{\mathrm{d}}{\mathrm{d}t}(\mathbf{I}\boldsymbol{\omega}) = \mathbf{I}\dot{\boldsymbol{\omega}} + \boldsymbol{\omega}\times(\mathbf{I}\boldsymbol{\omega}) \approx \mathbf{I}\dot{\boldsymbol{\omega}} \tag{13}$$

This approximation has been made in controllers such as [13], and discards the effect of precession and nutation of the rotating body. The $\boldsymbol{\omega}\times(\mathbf{I}\boldsymbol{\omega})$ term is small for bodies with small angular velocities and does not contribute significantly

to the dynamics of the robot. The inertia tensor in the world coordinate system can be found with

$$\mathbf{I} = \mathbf{R}\ {}_{\mathcal{B}}\mathbf{I}\ \mathbf{R}^{\mathsf{T}} \tag{14}$$

which, for small roll and pitch angles, can be approximated by

$$\hat{\mathbf{I}} = \mathbf{R}_z(\psi)\ {}_{\mathcal{B}}\mathbf{I}\ \mathbf{R}_z(\psi)^{\mathsf{T}} \tag{15}$$

where ${}_{\mathcal{B}}\mathbf{I}$ is the inertia tensor in body coordinates.

B. Simplified Robot Dynamics

The approximated orientation dynamics and translational dynamics can be combined into the following form:

$$\frac{\mathrm{d}}{\mathrm{d}t}\begin{bmatrix}\hat{\boldsymbol{\Theta}}\\ \hat{\mathbf{p}}\\ \hat{\boldsymbol{\omega}}\\ \hat{\dot{\mathbf{p}}}\end{bmatrix} = \begin{bmatrix}\mathbf{0}_3 & \mathbf{0}_3 & \mathbf{R}_z(\psi) & \mathbf{0}_3\\ \mathbf{0}_3 & \mathbf{0}_3 & \mathbf{0}_3 & \mathbf{1}_3\\ \mathbf{0}_3 & \mathbf{0}_3 & \mathbf{0}_3 & \mathbf{0}_3\\ \mathbf{0}_3 & \mathbf{0}_3 & \mathbf{0}_3 & \mathbf{0}_3\end{bmatrix}\begin{bmatrix}\hat{\boldsymbol{\Theta}}\\ \hat{\mathbf{p}}\\ \hat{\boldsymbol{\omega}}\\ \hat{\dot{\mathbf{p}}}\end{bmatrix} + \begin{bmatrix}\mathbf{0}_3 & \dots & \mathbf{0}_3\\ \mathbf{0}_3 & \dots & \mathbf{0}_3\\ \hat{\mathbf{I}}^{-1}[\mathbf{r}_1]_\times & \dots & \hat{\mathbf{I}}^{-1}[\mathbf{r}_n]_\times\\ \mathbf{1}_3/m & \dots & \mathbf{1}_3/m\end{bmatrix}\begin{bmatrix}\mathbf{f}_1\\ \vdots\\ \mathbf{f}_n\end{bmatrix} + \begin{bmatrix}0\\0\\0\\ \mathbf{g}\end{bmatrix} \tag{16}$$

Equation (16) can be rewritten with an additional gravity state to put the dynamics into the convenient state-space form:

$$\dot{\mathbf{x}}(t) = \mathbf{A}_c(\psi)\mathbf{x}(t) + \mathbf{B}_c(\mathbf{r}_1, ..., \mathbf{r}_n, \psi)\mathbf{u}(t) \tag{17}$$

where $\mathbf{A}_c \in \mathbb{R}^{13\times 13}$ and $\mathbf{B}_c \in \mathbb{R}^{13\times 3n}$. This form depends only on yaw and footstep locations. If these can be computed ahead of time, the dynamics become linear time-varying, which is suitable for convex model predictive control.

IV. Model Predictive Control

Desired ground reaction forces are found with a discrete-time finite-horizon model predictive controller. Because we consider ground reaction forces instead of joint torques, the predictive controller does not need to be aware of the configuration or kinematics of the leg. In general, model predictive controllers contain a model of the system to be controlled, and on each iteration, start from the current state and find the optimal sequence of control inputs and the corresponding state trajectory over the finite-length prediction horizon, subject to constraints on the state trajectory and the control inputs. Because this process is repeated for every iteration, only the first timestep of the computed control input trajectory is applied before the controller runs again and new control inputs are computed. In this section, we consider an MPC problem with horizon length k in the standard form

$$\min_{\mathbf{x},\mathbf{u}} \quad \sum_{i=0}^{k-1} ||\mathbf{x}_{i+1} - \mathbf{x}_{i+1,\text{ref}}||_{\mathbf{Q}_i} + ||\mathbf{u}_i||_{\mathbf{R}_i} \tag{18}$$

$$\text{subject to} \quad \mathbf{x}_{i+1} = \mathbf{A}_i\mathbf{x}_i + \mathbf{B}_i\mathbf{u}_i, i = 0\dots k-1 \tag{19}$$

$$\underline{\mathbf{c}}_i \le \mathbf{C}_i\mathbf{u}_i \le \overline{\mathbf{c}}_i, i = 0\dots k-1 \tag{20}$$

$$\mathbf{D}_i\mathbf{u}_i = 0, i = 0\dots k-1 \tag{21}$$

where $\mathbf{x}_i$ is the system state at time step i, $\mathbf{u}_i$ is the control input at time step i, $\mathbf{Q}_i$ and $\mathbf{R}_i$ are diagonal positive semidefinite matrices of weights, $\mathbf{A}_i$ and $\mathbf{B}_i$ represent the discrete time system dynamics, $\mathbf{C}_i, \underline{\mathbf{c}}_i$, and $\overline{\mathbf{c}}_i$ represent inequality constraints on the control input, and $\mathbf{D}_i$ is a matrix which selects forces corresponding with feet not in contact with the ground at timestep i. The notation $||\mathbf{a}||_{\mathbf{S}}$ is used to indicate the weighted norm $\mathbf{a}^{\mathsf{T}}\mathbf{S}\mathbf{a}$. The controller in this form attempts to find a sequence of control inputs that will guide the system along the trajectory $\mathbf{x}_{\text{ref}}$, trading off tracking accuracy for control effort, while obeying constraints. In the event that the system cannot exactly track the reference trajectory, which is often the case due to uncontrollable dynamics during periods of underactuation, or due to constraints, the predictive controller finds the best solution, in the least squares sense, over the prediction horizon. The optimization over the horizon while taking into account future constraints enables the predictive controller to plan ahead for periods of flight and regulate the states of the body during a gait when the body is always underactuated, such as bounding.

A. Force Constraints

The equality constraint in (21) is used to set all forces from feet off the ground to zero, enforcing the desired gait. The inequality constraint in (20) is used to set the following 10 inequality constraints for each foot on the ground

$$f_{\text{min}} \le f_z \le f_{\text{max}} \tag{22}$$

$$-\mu f_z \le \pm f_x \le \mu f_z \tag{23}$$

$$-\mu f_z \le \pm f_y \le \mu f_z \tag{24}$$

These constraints limit the minimum and maximum z-force as well as a square pyramid approximation of the friction cone.

B. Reference Trajectory Generation

The desired robot behavior is used to construct the reference trajectory. In our application, our reference trajectories are simple and only contain non-zero xy-velocity, xy-position, z position, yaw, and yaw rate. All parameters are commanded directly by the robot operator except for yaw and xy-position, which are determined by integrating the appropriate velocities. The other states (roll, pitch, roll rate, pitch rate, and z-velocity) are always set to 0. The reference trajectory is also used to determine the dynamics constraints and future foot placement locations. In practice, the reference trajectory is short (between 0.5 and 0.3 seconds) and recalculated often (every 0.05 to 0.03 seconds) to ensure the simplified dynamics remain accurate if the robot is disturbed.

C. Linear Discrete Time Dynamics

For each point n in the reference trajectory, an approximate $\hat{\mathbf{B}}_c[n] \in \mathbb{R}^{13\times 3n}$ matrix is computed from the $\mathbf{B}_c(\mathbf{r}_1, ..., \mathbf{r}_n, \psi)$ matrix defined in (17) using desired values of ψ and $\mathbf{r}_i$ from the reference trajectory and foot placement controller. Similarly, a single $\hat{\mathbf{A}}_c \in \mathbb{R}^{13\times 13}$ matrix is computed for the entire reference trajectory using the average value of ψ during the reference trajectory. The

$\mathbf{B}_c(\mathbf{r}_1, ..., \mathbf{r}_n, \psi)$ and $\mathbf{A}_c$ matrices are converted to a zero order hold discrete time model using the state transition matrix of the extended linear system:

$$\frac{\mathrm{d}}{\mathrm{d}t}\begin{bmatrix}\mathbf{x}\\ \mathbf{u}\end{bmatrix} = \begin{bmatrix}\mathbf{A} & \mathbf{B}\\ \mathbf{0} & \mathbf{0}\end{bmatrix}\begin{bmatrix}\mathbf{x}\\ \mathbf{u}\end{bmatrix} \tag{25}$$

This simplification allows us to express the dynamics in the discrete time form

$$\mathbf{x}[n+1] = \hat{\mathbf{A}}\mathbf{x}[n] + \hat{\mathbf{B}}[n]\mathbf{u}[n] \tag{26}$$

The approximation in (26) is only accurate if the robot is able to follow the reference trajectory. Large deviations from the reference trajectory, possibly caused by external or terrain disturbances, will result in $\hat{\mathbf{B}}[n]$ being inaccurate. However, for the first time step, $\hat{\mathbf{B}}[n]$ is calculated from the current robot state, and will always be correct. If, at any point, the robot is disturbed from following the reference trajectory, the next iteration of the MPC, which happens at most 40 ms after the disturbance, will recompute the reference trajectory based on the disturbed robot state, allowing it compensate for a disturbance.

D. QP Formulation

The optimization problem in (18) is reformulated to reduce problem size. While a solver that could take advantage of the sparsity and structure of our problem is likely to be the fastest solution, it was more straightforward and sufficiently fast to ignore the structure and sparsity and instead work on reducing the size of the problem. With a solver that does not exploit the sparsity of the problem, the formulation in equation (18) has a cubic time complexity with respect to horizon length, number of states (both control input and state trajectory), and number of constraints (both force and dynamics), so we achieve a significant speedup by removing the dynamics constraints and state trajectory from the constraints and optimization variables and include them in the cost function instead. This formulation also allows us to eliminate trivial optimization variables that are constrained to zero by (21) so that we are only optimizing forces for feet which are on the ground. This is based on the condensed formulation discussed in [28], but includes time-varying dynamics and state reference.

The condensed formulation allows the dynamics to be written as

$$\mathbf{X} = \mathbf{A}_{\text{qp}}\mathbf{x}_0 + \mathbf{B}_{\text{qp}}\mathbf{U} \tag{27}$$

where $\mathbf{X} \in \mathbb{R}^{13k}$ is the vector of all states during the prediction horizon and $\mathbf{U} \in \mathbb{R}^{3nk}$ is the vector of all control inputs during the prediction horizon.

The objective function which minimizes the weighted least-squares deviation from the reference trajectory and the weighted force magnitude is:

$$J(\mathbf{U}) = ||\mathbf{A}_{\text{qp}}\mathbf{x}_0 + \mathbf{B}_{\text{qp}}\mathbf{U} - \mathbf{x}_{\text{ref}}||_{\mathbf{L}} + ||\mathbf{U}||_{\mathbf{K}} \tag{28}$$

where $\mathbf{L} \in \mathbb{R}^{13k\times 13k}$ is a diagonal matrix of weights for state deviations, $\mathbf{K} \in \mathbb{R}^{3nk\times 3nk}$ is a diagonal matrix of weights for force magnitude, and $\mathbf{U}$ and $\mathbf{X}$ are the vectors of all control inputs and states during the prediction horizon. On the robot, we chose an equal weighting of forces $\mathbf{K} = \alpha\mathbf{1}_{3nk}$. The problem can be rewritten as:

$$\min_{\mathbf{U}} \quad \frac{1}{2}\mathbf{U}^\intercal\mathbf{H}\mathbf{U} + \mathbf{U}^\intercal\mathbf{g} \tag{29}$$

$$\text{s. t.} \quad \underline{\mathbf{c}} \le \mathbf{C}\mathbf{U} \le \overline{\mathbf{c}} \tag{30}$$

where $\mathbf{C}$ is the constraint matrix and

$$\mathbf{H} = 2(\mathbf{B}_{\text{qp}}^\intercal\mathbf{L}\mathbf{B}_{\text{qp}} + \mathbf{K}) \tag{31}$$

$$\mathbf{g} = 2\mathbf{B}_{\text{qp}}^\intercal\mathbf{L}(\mathbf{A}_{\text{qp}}\mathbf{x}_0 - \mathbf{y}) \tag{32}$$

The desired ground reaction forces are then the first $3n$ elements of $\mathbf{U}$. Notice that $\mathbf{H} \in \mathbb{R}^{3nk\times 3nk}$ and $\mathbf{g} \in \mathbb{R}^{3nk\times 1}$ both have no size dependence on the number of states, only number of feet n and horizon length k.

V. Results

TABLE I
Controller Settings and Robot Data

m	43 kg	$\mathbf{\Theta}$ weight	1
I_{xx}	0.41 kgm^2	z weight	50
I_{yy}	2.1 kgm^2	yaw rate weight	1
I_{zz}	2.1 kgm^2	$\mathbf{v}$ weight	1
μ	0.6	force weight (α)	1×10^{-6}
g_z	-9.8 m/s^2	$\mathbf{f}_{\text{min}}$	10 N
τ_{max}	250 Nm	$\mathbf{f}_{\text{max}}$	666 N

A. Experimental Setup

The proposed controller was implemented on the MIT Cheetah 3 Robot with a horizon length of one gait cycle (0.33 to 0.5 seconds), subdivided into between 10 and 16 timesteps, depending on the gait selected. New predictions were made at frequencies between 25 and 50 Hz, depending on the gait selected. Reference trajectories and desired gaits were generated based on inputs from a video game controller. Table I contains all parameters used for the predictive controller.

The result from the optimization is used directly and without filter in (4) to find joint torques. State estimation, swing leg planning, and leg impedance control happen at 1 kHz.

A simple heuristic function inspired by [1] is used to plan desired foot locations in the xy-plane of the world coordinate system:

$$\mathbf{p}^{\text{des}} = \mathbf{p}^{\text{ref}} + \mathbf{v}^{\text{CoM}}\Delta t/2 \tag{33}$$

where Δt is the time the foot will spend on the ground, $\mathbf{p}^{\text{ref}}$ is the corner of the rectangle described in Section II and $\mathbf{v}^{\text{CoM}}$ is the velocity of the robot's center of mass projected onto the xy-plane. The state estimator is used to determine these values for the current time step, and the reference trajectory is used for future values. The values of $\mathbf{p}^{\text{des}}$ are used for

For full videos, see `https://www.youtube.com/watch?v=q6zxCvCxhic`

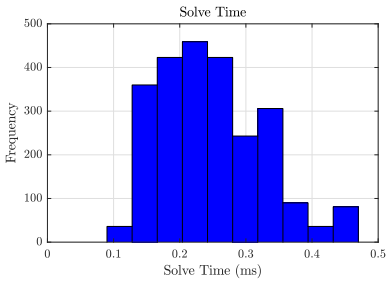

Fig. 4. Solve time during galloping

both the swing leg controller and to find $\mathbf{r}_i$ for the predictive controller.

The robot has an onboard computer with an Intel i7 laptop processor from 2011 and runs Ubuntu Linux with a kernel patched with CONFIG_PREEMPT_RT. The high level control code, including state estimation, swing leg control, and the dynamics calculations are implemented in MATLAB/Simulink and converted into C code. The MPC setup described in this paper is implemented in C++. The software uses the Eigen3 linear algebra library [29] and the qpOASES [24] quadratic programming library. The qpOASES solver specializes in solving quadratic programs for MPC with the online active set strategy [30] and was used to solve the convex optimization problem. See Figure 4 for a distribution of typical solve times.

To evaluate the predictive controller, it was tested with all of the common gaits: standing, trotting, a flying-trot, pacing, bounding, pronking, and galloping as well as an original 3-legged gait.

B. Trotting

A simple trotting gait was very effective at speeds under 1.2 m/s. While trotting, the robot has no preferred direction of travel; it can easily travel forward, backward, sideways, and along a diagonal at speed, or even travel in straight line while rotating around its center. Figure 5 demonstrates an aggressive turn in place with rotation speeds over 150 deg/sec, resulting in a linear foot velocity with respect to the body of over 1 m/s while the foot is on the ground. Additionally, the figure shows a separate aggressive acceleration test from -1 to 1 m/s in two seconds. Our ability to achieve these velocities on a 10 meter by 1.3 meter platform demonstrates the agility of the robot. The trotting gait was also tested outdoors at speeds up to 1.2 m/s, with the robot untethered. To demonstrate robustness, the robot was driven on muddy and sloped ground and remained stable, even when the feet slipped in the mud.

When trotting forward at high speed, feet in swing may collide with feet lifting off the ground, effectively limiting our trotting gait to low speeds. The flying-trot adds a flight period, improving the top speed to 1.7 m/s, and allows the robot to spin in place at 180 deg/s.

C. Disturbances and Stairs

To test the disturbance rejection of the controller, the robot was firmly kicked while trotting. The kick was forceful enough to accelerate the robot's body over 1 m/s sideways, but the robot was able to recover while keeping roll and pitch within 10 degrees of level, as shown in figure 7.

The flying trot gait was also tested on a staircase with 4 full size steps, which it was able to climb reliably. Approximately 15 pieces of wood simulating debris were scattered on the staircase to intentionally make the robot's feet slip. The swing trajectory height was increased to allow the feet to reach the next step, but the robot otherwise had no knowledge of the staircase. Contact detection algorithms [25] allowed the gait to be modified in real time as the feet touched down earlier or later than anticipated. We attempted to climb and descend the debris covered staircase 3 times. On the first attempt, the robot's front feet reached the top, but the operator mistakenly commanded a yaw rate of 120 deg/sec, causing a foot to step off the staircase, at which point the robot was shut off. The second attempt was successful at climbing and descending the stairs. In the third attempt, the robot again reached the top step and had a rear foot step off the staircase, causing the knee to invert. The robot slipped down a few stairs and regained control with its front feet on the first step. The operator commanded the robot to correct the orientation of the knee and tried again. This time, the robot was successful in ascending the stairs, but on the way down, as the front feet were about to step off the staircase, the robot kicked its own power switch and shut off. To the best of our knowledge, this is the first time a legged robot has climbed a staircase while using a dynamic gait with a flight period.

D. Pronking/Jumping

To demonstrate the robustness of our controller, we tested a pronking gait, which is a sequence of short jumps approximately 15 cm high. The robot spends 150 ms with all four feet on the ground, followed by 350 ms of flight. The robot is able to land and stabilize itself during the 150 ms of ground contact, allowing it to pronk in place indefinitely.

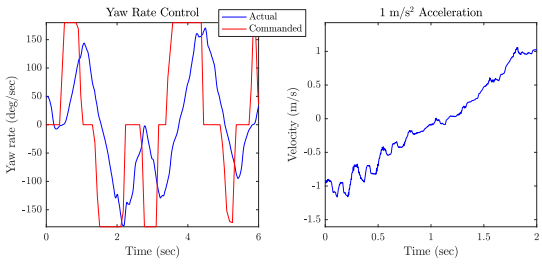

Fig. 5. Turning while trotting in place and separate acceleration test

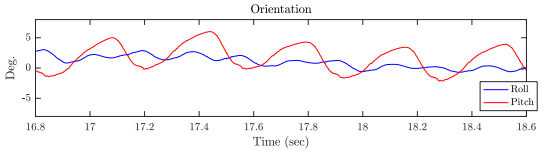

Fig. 6. Orientation control during 2.25 m/s bounding

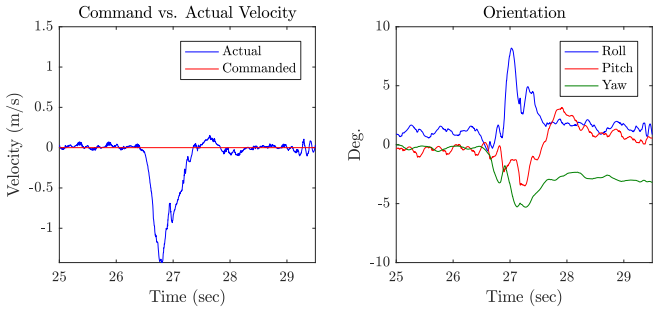

Fig. 7. Velocity and Orientation when the robot is kicked

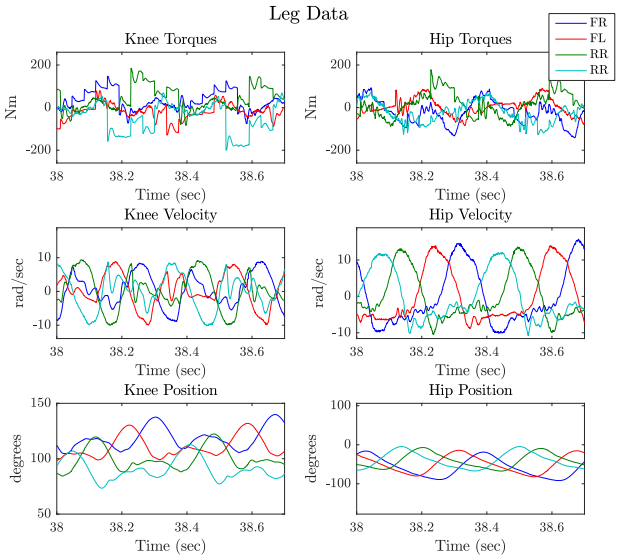

Fig. 8. Leg data during 2.5 m/s galloping

E. Bounding and Galloping

Bounding on Cheetah 3 is able to achieve higher speeds than trotting due to the higher gait frequency (3.3 Hz) and longer flight times. When bounding with low flight times, the legs must be narrowed in the front so that the rear feet can swing past. For our bounding gait, each foot is on the ground for 90 ms and in swing for 210 ms. While bounding, the robot was able to rotate in place at speeds up to 180 deg/sec. To demonstrate the stability of the controller, the spacing between the left and right feet was decreased until the robot fell. The robot was able to bound with the feet 10 cm apart indefinitely. With the feet 5 cm apart from each other, the robot was able to bound, but had poor control of sideways velocity, limiting the test to roughly 10 footsteps. Figure 6 shows the orientation during 2.25 m/s bounding. Bounding was tested up to 2.5 m/s and likely can go faster, however, galloping was observed to be much smoother, so it was used for all high speed tests. Galloping was able to reach a maximum speed of 3 m/s, at which point it became challenging to keep the robot on the treadmill. Figure 10 shows accurate velocity tracking of galloping up to the maximum speed, and figures 8 and 9 show leg and controller data during 2.5 m/s galloping.

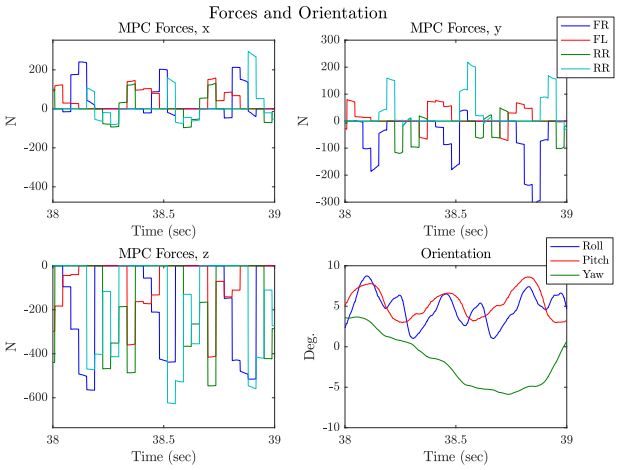

Fig. 9. Commanded forces and estimated orientation during 2.5 m/s galloping

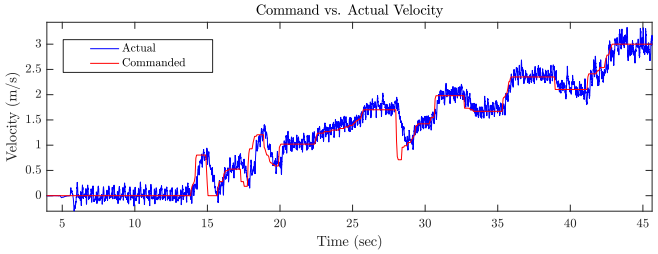

Fig. 10. Velocity control of galloping to 3 m/s

F. Limitations at High Speed

The MIT Cheetah 3 robot was developed with the intent to operate efficiently, carrying payloads indoors and near humans. As a result, its actuators and legs are designed for lower speeds than Cheetah 2, limiting the linear velocity of the feet. When testing high speed locomotion, the legs installed on Cheetah 3 had a maximum velocity of 15 rad/sec, significantly slower than the maximum of 24 rad/sec, observed on Cheetah 2. When operating near the hip velocity limit, the swing leg controller does not track well. In our attempts to go faster than 3 m/s, the legs could not swing far enough forward in time, causing the robot's orientation control to become less accurate. The robot was able to handle the increased roll and pitch error, but it became challenging to keep the robot on the 1.3 meter wide treadmill due to drift in yaw velocity. All of our high speed tests ended either with the robot accidentally being driven off the side of the treadmill or with a mechanical failure of the robot. We suspect that with increased leg velocity, higher speeds can be reached. In our full, multi-body simulation of the robot, we have achieved speeds up to 6 m/s when we increase leg velocity to a maximum of 30 rad/sec.

VI. Conclusions and Future Work

Despite significant simplifications to the robot dynamics model, the controller is shown to have good performance with a variety of gaits. We believe our controller is successful because our approximate model still captures many of

the important details of locomotion, including the ground reaction force constraints. We find that a highly accurate model of the robot's dynamics during the prediction horizon is less important than an accurate model of the robot's instantaneous dynamics. Our approach updates the dynamics during the prediction horizon frequently and with minimal delay, ensuring the instantaneous approximation is always accurate. Between iterations of our control algorithm, a linear approximation of dynamics appears to be sufficient.

In the future, we plan to more thoroughly investigate different gaits and aggressive behaviors. It is very likely that higher speeds and greater stability can be achieved by designing a more favorable gait. So far, all of the gaits used with the proposed controller were created manually, and imitated gaits found in dogs and other legged robots. We plan to replace these predetermined gaits with contact patterns determined by a higher level planner to improve the robustness and performance of our locomotion. This framework would allow the higher-level planner to run slowly and the low-level predictive controller can continue to run rapidly.

REFERENCES

[1] M. H. Raibert, *Legged Robots That Balance*. Cambridge, MA, USA: Massachusetts Institute of Technology, 1986.

[2] H.-W. Park, P. M. Wensing, and S. Kim, "High-speed bounding with the mit cheetah 2: Control design and experiments," *The International Journal of Robotics Research*, vol. 36, no. 2, pp. 167–192, 2017.

[3] D. P. Krasny and D. E. Orin, "Evolution of a 3d gallop in a quadrupedal model with biological characteristics," *Journal of Intelligent & Robotic Systems*, vol. 60, no. 1, pp. 59–82, 2010.

[4] R. Orsolino, M. Focchi, D. G. Caldwell, and C. Semini, "A combined limit cycle - zero moment point based approach for omni-directional quadrupedal bounding," in *International Conference on Climbing and Walking Robots (CLAWAR)*, 2017.

[5] C. Gehring, S. Coros, M. Hutler, C. D. Bellicoso, H. Heijnen, R. Diethelm, M. Bloesch, P. Fankhauser, J. Hwangbo, M. Hoepflinger, and R. Siegwart, "Practice makes perfect: An optimization-based approach to controlling agile motions for a quadruped robot," *IEEE Robotics Automation Magazine*, vol. 23, no. 1, pp. 34–43, March 2016.

[6] M. Rutschmann, B. Satzinger, M. Byl, and K. Byl, "Nonlinear model predictive control for rough-terrain robot hopping," in *2012 IEEE/RSJ International Conference on Intelligent Robots and Systems*, Oct 2012, pp. 1859–1864.

[7] J. Koenemann, A. D. Prete, Y. Tassa, E. Todorov, O. Stasse, M. Bennewitz, and N. Mansard, "Whole-body model-predictive control applied to the hrp-2 humanoid," in *2015 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, Sept 2015, pp. 3346–3351.

[8] Y. Tassa, T. Erez, and E. Todorov, "Synthesis and stabilization of complex behaviors through online trajectory optimization," in *2012 IEEE/RSJ International Conference on Intelligent Robots and Systems*, Oct 2012, pp. 4906–4913.

[9] G. Bledt, P. M. Wensing, and S. Kim, "Policy-regularized model predictive control to stabilize diverse quadrupedal gaits for the mit cheetah," in *2017 IEEE International Conference on Intelligent Robots and Systems*, 2017.

[10] A. W. Winkler, D. C. Bellicoso, M. Hutter, and J. Buchli, "Gait and trajectory optimization for legged systems through phase-based end-effector parameterization," *IEEE Robotics and Automation Letters (RA-L)*, may 2018.

[11] A. W. Winkler, F. Farshidian, D. Pardo, M. Neunert, and J. Buchli, "Fast trajectory optimization for legged robots using vertex-based zmp constraints," *IEEE Robotics and Automation Letters (RA-L)*, vol. 2, pp. 2201–2208, oct 2017.

[12] M. Neunert, M. Stäuble, M. Giftthaler, C. D. Bellicoso, J. Carius, C. Gehring, M. Hutter, and J. Buchli, "Whole-Body Nonlinear Model Predictive Control Through Contacts for Quadrupeds," *ArXiv e-prints*, Dec. 2017.

[13] M. Focchi, A. del Prete, I. Havoutis, R. Featherstone, D. G. Caldwell, and C. Semini, "High-slope terrain locomotion for torque-controlled quadruped robots," *Autonomous Robots*, vol. 41, no. 1, pp. 259–272, Jan 2017.

[14] Y. Abe, M. da Silva, and J. Popović, "Multiobjective control with frictional contacts," in *Proceedings of the 2007 ACM SIGGRAPH/Eurographics Symposium on Computer Animation*, ser. SCA '07. Aire-la-Ville, Switzerland, Switzerland: Eurographics Association, 2007, pp. 249–258.

[15] B. J. Stephens and C. G. Atkeson, "Push recovery by stepping for humanoid robots with force controlled joints," in *2010 10th IEEE-RAS International Conference on Humanoid Robots*, Dec 2010, pp. 52–59.

[16] A. Ruina, J. E. Bertram, and M. Srinivasan, "A collisional model of the energetic cost of support work qualitatively explains leg sequencing in walking and galloping, pseudo-elastic leg behavior in running and the walk-to-run transition," *Journal of theoretical biology*, vol. 237, no. 2, pp. 170–192, 2005.

[17] R. M. Walter and D. R. Carrier, "Rapid acceleration in dogs: ground forces and body posture dynamics," *Journal of Experimental Biology*, vol. 212, no. 12, pp. 1930–1939, 2009.

[18] J. A. Smith and I. Poulakakis, "Rotary gallop in the untethered quadrupedal robot scout ii," in *2004 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS) (IEEE Cat. No.04CH37566)*, vol. 3, Sept 2004, pp. 2556–2561 vol.3.

[19] H.-W. Park and S. Kim, "Quadrupedal galloping control for a wide range of speed via vertical impulse scaling," *Bioinspiration and Biomimetics*, vol. 10, no. 2, p. 025003, 2015.

[20] "Introducing WildCat," https://www.youtube.com/watch?v=wE3fmFTtP9g, Oct 2013.

[21] S. Boyd and L. Vandenberghe, *Convex optimization*. Cambridge university press, 2004.

[22] Y. Wang and S. Boyd, "Fast model predictive control using online optimization," *IEEE Transactions on Control Systems Technology*, vol. 18, no. 2, pp. 267–278, March 2010.

[23] A. Domahidi, E. Chu, and S. Boyd, "ECOS: An SOCP solver for embedded systems," in *European Control Conference (ECC)*, 2013, pp. 3071–3076.

[24] H. Ferreau, C. Kirches, A. Potschka, H. Bock, and M. Diehl, "qpOASES: A parametric active-set algorithm for quadratic programming," *Mathematical Programming Computation*, vol. 6, no. 4, pp. 327–363, 2014.

[25] G. Bledt, P. M. Wensing, S. Ingersoll, and S. Kim, "Contact model fusion for event-based locomotion in unstructured terrains," in *2018 IEEE International Conference on Robotics and Automation*, 2018.

[26] P. M. Wensing, A. Wang, S. Seok, D. Otten, J. Lang, and S. Kim, "Proprioceptive actuator design in the mit cheetah: Impact mitigation and high-bandwidth physical interaction for dynamic legged robots," *IEEE Transactions on Robotics*, vol. 33, no. 3, pp. 509–522, June 2017.

[27] J. Craig, *Introduction to Robotics: Mechanics and Control*, ser. Addison-Wesley series in electrical and computer engineering: control engineering. Pearson/Prentice Hall, 2005.

[28] J. L. Jerez, E. C. Kerrigan, and G. A. Constantinides, "A condensed and sparse qp formulation for predictive control," in *2011 50th IEEE Conference on Decision and Control and European Control Conference*, Dec 2011, pp. 5217–5222.

[29] G. Guennebaud, B. Jacob *et al.*, "Eigen v3," http://eigen.tuxfamily.org, 2010.

[30] H. Ferreau, H. Bock, and M. Diehl, "An online active set strategy to overcome the limitations of explicit mpc," *International Journal of Robust and Nonlinear Control*, vol. 18, no. 8, pp. 816–830, 2008.