2019 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)
Macau, China, November 4-8, 2019

Implementing Regularized Predictive Control for Simultaneous Real-Time Footstep and Ground Reaction Force Optimization

Gerardo Bledt[1] and Sangbae Kim[1]

Abstract— **This work presents a successful implementation of a nonlinear optimization-based Regularized Predictive Control (RPC) for legged locomotion on the MIT Cheetah 3 robot platform. Footstep placements and ground reaction forces at the contact feet are simultaneously solved for over a prediction horizon in real-time. Often in academic literature not enough attention is given to the implementation details that make the theory work in practice and many times it is precisely these details that end up being critical to the success or failure of the theory in real world applications. Nonlinear optimization for real-time legged locomotion control in particular is one of the techniques that has shown promise, but falls short when implemented on hardware systems subjected to computation limits and undesirable local minima. We discuss various algorithms and techniques developed to overcome some of the challenges faced when implementing nonlinear optimization-based controllers for dynamic legged locomotion.**

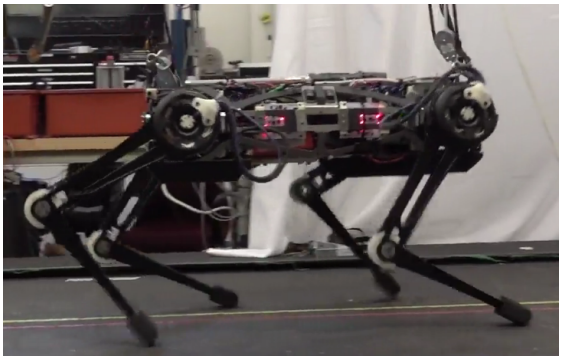


Fig. 1: **MIT Cheetah 3 Locomotion.** The RPC optimization finds ground reaction forces and footstep locations in real-time for realizing dynamic legged locomotion.

I. Introduction

Research on legged robots has seen a recent growth in interest due to advances demonstrating highly dynamic locomotion and ability to traverse difficult and unstructured terrains. This is in part because of developments resulting in more sophisticated controllers capable of dealing with increasingly difficult situations, as well as improved computing and actuation methods capable of robustly executing these controllers in the real world. Optimization-based controllers for legged robots have been successful in balancing, locomotion, and disturbance rejection. However, they are still lacking when attempting to solve more complex problems due to convergence and timing difficulties.

Legged robots currently use a variety of optimization-based locomotion controllers. The Atlas humanoid robot uses optimization throughout its control architecture [1]. HyQ at IIT uses a simple template model in a whole-body controller for instantaneous optimization of ground reaction forces [2] with heuristic planning to modify the desired inputs and deal with the aspects of locomotion that do not directly contribute to force controlled balance such as swing leg reflex [3]. ANYmal from ETH Zürich has implemented a hierarchical whole-body controller to execute tasks in order of priority for instantaneous force control of dynamic gaits [4], [5].

Instantaneous whole-body controllers have the limitation of only being aware of the current state rather than having knowledge about the gait and upcoming states meaning that they cannot easily control gaits with flight or longer periods of underactuation. For this reason, a heavy focus has been placed on predictive controllers, with Model Predictive Control (MPC) being the main framework of choice. MPC has been shown for many years to be theoretically successful in simulation [6], [7], [8]. However, robust hardware implementations are only recently beginning to show good performance. A whole-body MPC implementation was shown on the HRP-2 humanoid platform [9]. In this work, the authors state many of their solutions to address the difficulties in whole-body MPC, namely the computational cost, undesirable local minima, and discontinuities in control.

MPC is commonly used as a higher level planner rather than as the primary stabilizing controller. Many of these techniques solve the optimization for a large number of future footsteps and then another controller tracks the plan online. Work by Winkler et al. generates footholds and CoM trajectories for a certain look ahead period and then has a separate controller track the motion in practice [10], [11]. These fast techniques for simultaneous foothold and CoM planning have been shown on hardware, but are geared towards planning for execution rather than real-time, coupled reactive adaptation.

Real-time Nonlinear MPC (NMPC) techniques are gaining popularity with the improvements in computing capabilities. Quadcopters have shown success in fast implementations of NMPC [12], [13]. In legged robots, the LOLA robot uses real-time NMPC to adapt its footstep placements online under disturbances [14]. Farshidian et al. demonstrate their FastSLQ-MPC for generating a real-time whole-body nonlinear MPC to create trotting gaits [15]. A real-time whole-body NMPC implementation is shown on hardware in [16] where the robot is able to stand and do slow trotting. However, they note that computation and real-world limitations must still be overcome to take full advantage of the nonlinear optimization's capabilities.

[1]Department of Mechanical Engineering, MIT, Cambridge, MA 02139, USA: gbledt@mit.edu, sangbae@mit.edu

In this paper, the controller is designed for the MIT Cheetah 3 robot platform pictured in Figure 1. Two separate balance controllers have been implemented on the robot. The first being a Quadratic Program (QP) based balance controller that uses instantaneous ground reaction forces at the contact points to regulate the CoM states [17]. The second being a linear, convex MPC implementation which is able to stabilize various dynamic gaits as described in [18]. Both assume fixed footstep locations. Previous work showed promising results with a similar quadruped model using nonlinear optimization to pick both footsteps and ground reaction forces in simulation [19]. This paper presents the Regularized Predictive Control (RPC) hardware implementation as an extension of that work relying on heuristic regularization rather than a high-fidelity dynamics model.

A. Contribution

The contribution of this paper lies with the culmination of various important research efforts into a functional hardware implementation. As pointed out, the system integration of a nonlinear optimization-based controller is non-trivial and as such it is not yet widely used as the main real-time controller in legged robots. This work is intended to present some of the more critical details that led to the successful implementation of the RPC framework for real-time control of a dynamic quadruped. The hope is that other NMPC-style methods can benefit and overcome the computation limitations that currently plague real-time nonlinear optimization for control. The results show the nonlinear predictive controller is able to perform as adequately or better than the previous controllers on the robot during the experiments. We believe that this is a big step in showing that nonlinear predictive controllers are approaching a point where they can be used robustly to control dynamic legged robots.

The following sections are organized as follows. Section II presents the nonlinear RPC optimization framework with the cost function and constraints as written in the theory. Then Section III presents a survey of the implementation details that allow the theoretical controller to be implemented on the hardware in real-time. Results are shown for various experimental situations in Section IV. Finally, Section V discusses brief conclusions of the work.

II. Nonlinear RPC Optimization Framework

The RPC framework described here is an extension of previous work by the authors in [19]. It was shown that adding heuristic regularization directly into the optimization allowed the controller to exploit the known dynamics of the system resulting in faster computation time and better conditioned cost spaces. However, that version was presented purely in simulation where time could be stopped while the optimization searched for a solution. This is not realistic on the actual hardware and so while the theoretical controller and optimization framework have not been modified significantly, the implementation details are drastically different.

The control model uses massless leg and linear CoM dynamics assumptions to simplify some of the nonlinearities

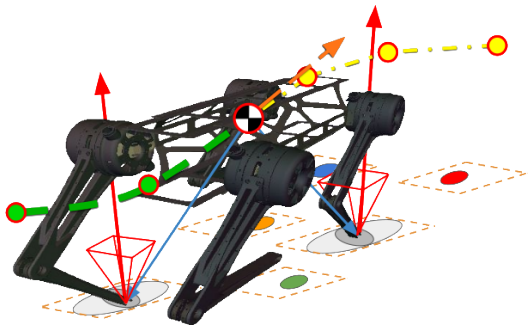


Fig. 2: **RPC Optimization.** Physically realistic constraints are enforced for the simplified dynamics, footsteps locations, and ground reaction forces.

and make the dynamics calculation along with the associated gradients simple to solve. This allows us to use the dynamics as a rough feasibility constraint rather than a strict dominating condition. The optimization framework with its cost function and explicitly written constraints is posed as

$$\min_{\boldsymbol{x},\boldsymbol{u}} \quad \sum_{k=0}^{N-1} (\underbrace{\tilde{\boldsymbol{x}}_k^T \boldsymbol{Q}_k \tilde{\boldsymbol{x}}_k}_{\text{Dynamics}} + \underbrace{\tilde{\boldsymbol{u}}_k^T \boldsymbol{R}_k \tilde{\boldsymbol{u}}_k}_{\text{Regularization}}) + \underbrace{\tilde{\boldsymbol{x}}_N^T \boldsymbol{Q}_N \tilde{\boldsymbol{x}}_N}_{\text{Terminal State}}$$

subject to

Simplified Discrete Dynamics
$$\boldsymbol{x}_{k+1} - (\boldsymbol{A}_k \boldsymbol{x}_k + \boldsymbol{B}_k h(\boldsymbol{x}_k, \boldsymbol{u}_k) + \boldsymbol{d}_k) = 0$$

Foot Placed on Ground
$$\hat{z}_g(p_{x,i}, p_{y,i}) - p_{z,i} = 0$$

Kinematic Leg Limits
$$\left\| \boldsymbol{r}_{ref,i} - \boldsymbol{r}_{h,i} \right\| - z_0 \tan(\tfrac{\pi}{6}) \leq 0$$

Foot Stationary During Stance
$$\boldsymbol{s}_{\phi,i,k+1} \boldsymbol{s}_{\phi,i,k} \left(\boldsymbol{p}_{i,k+1} - \boldsymbol{p}_{i,k} \right) = 0$$

Positive Ground Force Normal
$$-\hat{\boldsymbol{f}}_i \cdot \nabla \hat{\mathbb{G}}(\hat{\boldsymbol{p}}_i) \leq 0$$

Lateral Force Friction Pyramids
$$-\mu \boldsymbol{f}_{z,i} \leq \boldsymbol{f}_{x,i} \leq \mu \boldsymbol{f}_{z,i}$$
$$-\mu \boldsymbol{f}_{z,i} \leq \boldsymbol{f}_{y,i} \leq \mu \boldsymbol{f}_{z,i}$$

The robot states are the CoM position and Euler angles and their derivatives, $\boldsymbol{x} = \begin{bmatrix} \boldsymbol{p}^T & \boldsymbol{\Theta}^T & \dot{\boldsymbol{p}}^T & \dot{\boldsymbol{\Theta}}^T \end{bmatrix}^T$. The inputs are the foot positions relative to the CoM and their ground reaction forces, $\boldsymbol{u} = \begin{bmatrix} \boldsymbol{r}_1^T & \boldsymbol{f}_1^T \dots \boldsymbol{r}_4^T & \boldsymbol{f}_4^T \end{bmatrix}^T$. Decision variables for each timestep, k, include both the states and inputs, $\boldsymbol{\chi}_k = \begin{bmatrix} \boldsymbol{x}_k^T & \boldsymbol{u}_k^T \end{bmatrix}^T$. Scheduled gait phase for each foot, $\boldsymbol{s}_{\phi,i}$, foot position on the ground, $\boldsymbol{p}_i$, and estimated ground height, $\hat{z}(p_{x,i}, p_{y,i})$, are used for the foot constraints.

The framework enforces the necessary constraints for physical realizability characteristic of a point foot legged system as depicted in Figure 2. For example the forces are constrained so that the feet cannot pull on the ground. However, we make the assumption that the feet will not slip

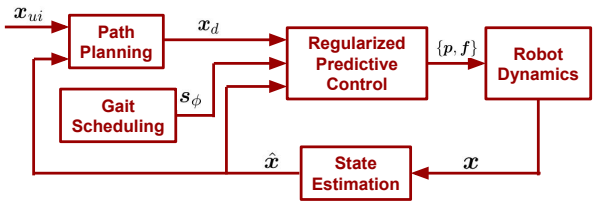


Fig. 3: **Simplified Control Framework.** Simplified block diagram showing the general flow of information throughout the control framework. In theory, the RPC block is simply the optimization with no modifications needed.

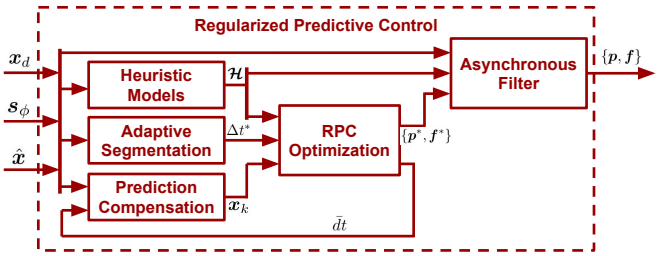


Fig. 4: **Modified RPC Implementation.** The theoretical RPC optimization is aided by various other algorithms to overcome some of the limitations and difficulties that come with real world implementation.

on the ground and write it as a constraint even though it is possible for the feet to slip. We attempt to avert this by adding the friction constraint to avoid large lateral forces. Although many of the terrain conditions are explicitly written in the constraints, it is worth noting that the hardware experiments in this paper are conducted through blind locomotion, meaning that the robot has no external perception of the environment. Therefore, assumptions about the terrain are made for the optimization constraints.

While regularization is often used in optimal control, this method attempts to regularize to a time-varying, nonlinear function composed of heuristics designed to closely approximate the optimal solution in various operating situations. When implemented on the hardware, the choice and design of the heuristic regularization became much more important than the actual dynamics model being optimized over. Solutions perform well towards the beginning of the prediction horizon, but breakdown towards the later parts of a long horizon as the dynamics are not highly accurate to the system. For this reason, it is desirable to constantly replan as fast as possible.

III. Implementation Details

The theory behind the controller has been shown in simulation, but robotics requires engineering to actually implement. While in Figure 3 the overall block diagram is presented with a single block representing the theoretically derived RPC, in practice it is not so simple to just run the optimization assuming that it will be solved instantaneously. Figure 4 extends on the theoretical derivation and shows the algorithms and modifications designed to aide the optimization overcome the real-time implementation challenges. This section describes the blocks in the diagram and their function in the control framework resulting in successful locomotion on the hardware platform.

A lot of the success comes from managing the tradeoff between accuracy and computation time, as well as preprocessing the Nonlinear Program (NLP) inputs by exploiting prior knowledge of the system. The MIT Cheetah 3 robot is designed to be robust to inaccuracies in ground reaction forces and has been shown to be adequately modeled for control with massless legs. This affords the ability to use the simple dynamics model in the optimization. A one time per solution preprocessing calculation has powerful effects on conditioning the problem's initial guesses and cost landscape.

A. Direct Transcription

Various optimization methods are available for solving optimal control problems. The choice of which has large effects on the solution accuracy and speed with each having their own benefits and drawbacks. For the controller we decided to use direct transcription which uses piecewise constant input and a piecewise linear state trajectories with both the states and the inputs used as decision variables in the optimization [20].

By posing the optimization as a direct transcription we lose some of the accuracy we may gain using a more sophisticated method, but also reduce the computation times. The loss in accuracy is not critical since the model we are using is an approximation rather than high-fidelity. As such, tolerances and convergence conditions can be greatly relaxed. It is not imperative to solve for a high degree of optimality on a model that is already a large approximation. We found that while the predicted robot states further along the prediction horizon rarely match with the actual states at that future time, the robot states generally trended towards the predicted directions. However, since the controller is constantly replanning, new inputs are modified to account for the discrepancies arising from the coarseness and simplicity of the optimization model dynamics.

With direct transcription the dynamics constraints are only dependent on the current timestep and the next timestep. This implicitly constrains the dynamics to be enforced throughout the prediction rather than explicitly adhering to the dynamics from the first to the last timestep. Similarly, even though footsteps may extend over several timesteps, by only constraining adjacent stance positions the constraint does not need to be explicitly written over all the relevant timesteps. This removes a lot of redundancies and better conditions the problem to be a series of small linear approximations linked together to approximate the overall nonlinear function. Only each small step needs to be constrained to be dynamically feasible rather than the full, highly nonlinear function. This results in more accurate dynamics between timesteps as the linear dynamics hold more fidelity to the real nonlinear dynamics over smaller time intervals.

Consequently, Figure 5 shows the sparsity matrices for the Constraint Jacobian (5a) and the Hessian of the Lagrangian

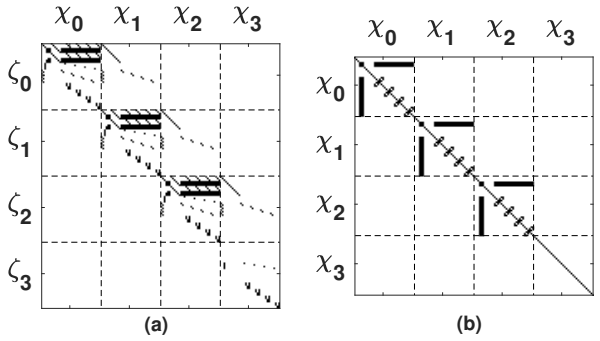


Fig. 5: **Sparsity Patterns.** Both the constraint Jacobian (a) and Lagrangian Hessian (b) feature a highly sparse, repeated structure throughout the matrix which contributes to the flexibility and speed of the algorithm.

(5b) where the dark spots indicate the indices in the matrices where the entries are possible to be non-zero. $\boldsymbol{\chi}$ signifies the decision variables in the optimization, $\boldsymbol{\zeta}$ is the set of constraints, while the subscripts denote the prediction timestep number. In this example $N = 4$, but in practice it can be chosen to be anything. By design, the matrices are both sparse and although the size of the Constraint Jacobian and Lagrangian Hessian grow proportional to N^2, the non-zero entries of each grows linearly as, $\mathcal{O}(N)$.

All parts of the equations and constraints that relate to contact points are defined by the contact state boolean variable, $\boldsymbol{s} = \{0 = swing, 1 = contact\}$. This provides a simple way to automatically adapt the active decision variables for each contact mode without having to explicitly write a different optimization for each combination of foot contacts. This propagates through the prediction horizon and activates or deactivates different contact modes in the gradients and Hessians according to the gait schedule regardless of the order or duration of each.

Algorithm 1: Adaptive Timestep Segmentation

nominal timestep: $\Delta t_{nominal}$;
scheduled contact to swing phase: $\phi_{c\to\bar{c}} = \frac{T_p - T_{\bar{c}}}{T_p}$;
for $k = 0$ *to NUM_PREDICTIONS-1* **do**
  number of feet in swing: $N_{\bar{c}} = 0$;
  $\Delta t_{max} = \infty$
  for $f = 0$ *to NUM_FEET-1* **do**
    if $\phi_{f,k} < \phi_{c\to\bar{c}}$ **then**
      foot in contact: $s_{f,k} = 1$;
      time until foot switches to swing: $\Delta t_{switch} = T_p(\phi_{c\to\bar{c}} - \phi_{f,k})$;
    else
      foot out of contact: $s_{f,k} = 0$;
      time until foot switches to contact: $\Delta t_{switch} = T_p(1 - \phi_{f,k})$;
      $N_{\bar{c}}$++;
    if $\Delta t_{switch} < \Delta t_{max}$ **then**
      $\Delta t_{max} = \Delta t_{switch}$;
  choose the current segment duration: Δt_k^*;
  if $N_{\bar{c}}$ = *NUM_FEET* **then**
    $\Delta t_k^* = \Delta t_{max}$;
    $flight\_phase = 1$;
  else if $\Delta t_{max} < \Delta t_{nominal}$ **then**
    $\Delta t_k^* = \Delta t_{max}$;
    $flight\_phase = 0$;
  else
    nominally divide time till switch evenly: $\Delta t_k^* = \frac{\Delta t_{max}}{round(\frac{\Delta t_{max}}{\Delta t_{nominal}})}$;
    $flight\_phase = 0$;
  $\phi_{f,k+1} = \text{mod}(\phi_{f,k} + \frac{\Delta t_k^*}{T_P}, 1)$;

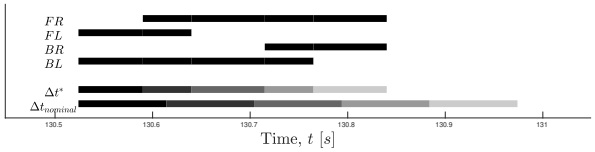
B. Adaptive Timing Segmentation

Since Legged locomotion is defined by the discrete changes between contact and non-contact modes of each of the legs, the optimization timestep segmentations must reflect these modes. As the controller is run continuously rather than at a scheduled time, the time step vector to be optimized over cannot simply be prescribed. As such, an algorithm must be developed to adaptively segment the time vector regardless of the scheduled gait pattern.

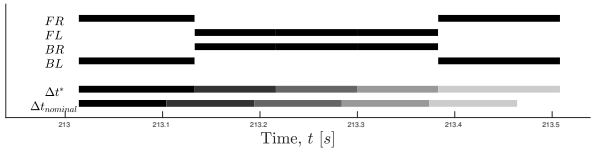
The algorithm takes a nominal discretization timestep sequence and attempts to calculate the closest set of timestep segmentations while respecting the gait schedule. Discrete scheduled contact switches must signify a new division as the robot enters a new control mode. Since this often happens irregularly, the algorithm chooses the number of timesteps within the same contact mode and spaces them out evenly. Algorithm 1 outlines this process. This allows for an incredible amount of freedom in choosing gait patterns and frequencies. Instead of constraining the predicted timestep vector to be divided evenly, any gait schedule can be optimized over without needing to make any change in the controller.

The ambling gait in Figure 6a has 8 quick contact switches over the full gait cycle and therefore the algorithm shortens the overall prediction horizon. The trotting gait in Figure 6b has 2 contact switches over the nominal horizon, since diagonal pairs of legs switch simultaneously and can be captured more easily by the nominal prediction horizon. For those more complex gaits, it is easy to simply increase the number of prediction discretizations.

This algorithm is $\mathcal{O}(N)$, so it is straight-forward and computationally inexpensive to add checks for various boolean flags that get used in the calculations. For example, if a foot is determined to be in contact, we can check if it was previously in contact or not, which signifies a touchdown event. Keeping track of touchdown events lets the optimization know if it should use the heuristic reference footstep location or the footstep location from the previous iteration during initialization. The no foot slip constraints will be enforced either way so it is not necessary to explicitly flag touchdown events,

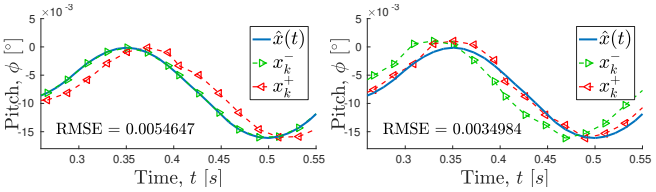


(a) Adapted Amble Gait Segmentation



(b) Adapted Trot Gait Segmentation

Fig. 6: **Adaptive Timestep Segmentation.** Top four bars represent the upcoming gait scheduled contact sequences for each of the feet. The bottom bars are the nominal timestep segmentation and the adapted segmentation based on contact mode switches where each segment gets lighter further along the prediction horizon.



(a) No Prediction (b) With Prediction

Fig. 7: **Predictive Delay Compensation.** Blue trajectory shows the actual robot states, while the green and red trajectories show the initial state at the beginning and end of the optimization respectively.

but initializing and regularizing with already correct footstep logic decreases the number of iterations and complexity needed to converge.

C. Prediction Delay Compensation

The optimization solve time is slower than the $1kHz$ main control loop so there are often timing delays resulting in discrepancies between the current state of the robot with its associated scheduled contact mode and the returned optimal solution. Forces and step locations are applied after a solution is returned, so the initial optimization states at the beginning and end of the solve time, dt^*, are equal, $\boldsymbol{x}_0^- = \boldsymbol{x}_0^+$, but the actual robot state may change, $\hat{\boldsymbol{x}}(t) \neq \hat{\boldsymbol{x}}(t + dt^*)$ Therefore, we must account for the fact that the robot is likely not in the same state as when the optimization began and that legs may have switched into a different contact state.

As such, the initial state given to the solver, $\boldsymbol{x}_0$ is not the current instantaneous state of the robot, but rather a predicted future state that the robot can be expected to be near when a solution is returned as solved for by the linear dynamics approximation

$$\boldsymbol{x}_0 = \boldsymbol{A}(\bar{dt})\hat{\boldsymbol{x}}(t) + \boldsymbol{B}(\bar{dt})h(\hat{\boldsymbol{x}}(t), \boldsymbol{u}(t)) + \boldsymbol{d}(\bar{dt}) \quad (1)$$

where $\boldsymbol{u}(t)$ is the current input from the last solution and the dynamics are integrated forward using $\bar{dt}$, the filtered average optimization solve time.

Similarly, the phase of each foot will be different by the time the optimization returns a value and therefore may also be scheduled to be in a different contact state than it is currently in. Each leg phase can be modified according to

$$\bar{d\phi} = \frac{\bar{dt}}{T_P} \quad (2)$$

$$\phi_k = \text{mod}(\phi(t) + \bar{d\phi}, 1) \quad (3)$$

where T_P is the nominal gait period time and $\phi(t)$ is the current phase of the foot. Again, this starts the prediction with the contact mode that the robot will likely be in when a solution is returned. If a foot is nearing the end of stance, it is not a good idea to solve for forces assuming that the foot will be in stance when the optimization converges. If there is a mode change before the end, then whatever solution was found is no longer possible to execute. Figure 7 shows that the Root Mean Squared Error (RMSE) between the current robot state, $\boldsymbol{x}(t)$, and initial optimization state when a solution is returned, $\boldsymbol{x}_0^+$, is lowered by using prediction.

D. Extracted Heuristic Models

In previous work it was demonstrated that the performance of the controller could improve drastically simply by adding improved heuristic regularization to the RPC. Particularly since most NLP solvers are designed for smooth continuous problems and legged locomotion is by definition a discrete hybrid problem, the regularization heuristics help guide and attract the solutions towards known operating regions. Discontinuities can prevent the solution from converging or finding local minima so the regularization warps the cost space in a favorable way to find a solution quickly.

In the majority of situations, the heuristics for desired CoM state, foot locations, and ground reaction forces provide a "good enough" solution for locomotion. By simply executing the heuristics, the robot should be able to remain upright for at least a small amount of time. This gives the RPC enough time to return a new solution without falling over. The heuristics are embedded directly into the optimization through the error term in the quadratic cost functions as

$$\tilde{\boldsymbol{x}} = \boldsymbol{\mathcal{H}}_x(\hat{\boldsymbol{x}}, \boldsymbol{x}_d) - \boldsymbol{x}_k \quad (4)$$

$$\tilde{\boldsymbol{u}} = \boldsymbol{\mathcal{H}}_u(\hat{\boldsymbol{x}}, \boldsymbol{x}_d) - \boldsymbol{u}_k \quad (5)$$

This regulates the states and inputs towards the designed heuristics rather than the simple user input commands, which act as a higher level goal for the robot, but are modified with the heuristics in order to better solve the optimization.

Designing these heuristics can be challenging and outside the scope of this work. For the purposes of this paper, we will assume that a set of heuristics, $\boldsymbol{\mathcal{H}}$, is pre-determined. A heuristic extraction framework for determining simple, physically meaningful models from data analysis as well as expert design will be presented in future work. The results of adding heuristic regularization models showed a roughly

$\sim 2\times$ improvement in maximum forward, lateral, and turning velocities simply by exploiting the heuristics. No changes were made to the control gains or the controller structure to incite the performance boost.

E. Asynchronous Solution Filtering

The prediction delay compensation in Section III-C starts the optimization acting on a future state and contact mode based on the moving average of the optimization solution time. However, this means that if the actual gait schedule has a differing contact mode than the resulting RPC solution, there must be a way to deal with this discrepancy. For both the footstep and ground reaction forces of each leg, the nominal values are given to be the extracted heuristics discussed in Section III-D. As stated, the heuristics are designed precisely for this purpose. Therefore in the event that the optimization does not return a solution or cannot converge before the contact mode changes, the robot simply executes the heuristic and is able to temporarily get away without a new optimization solution. This temporary control action is designed for idealized conditions and will not stabilize the robot indefinitely. However, it will provide a "good enough" solution to avoid a fall.

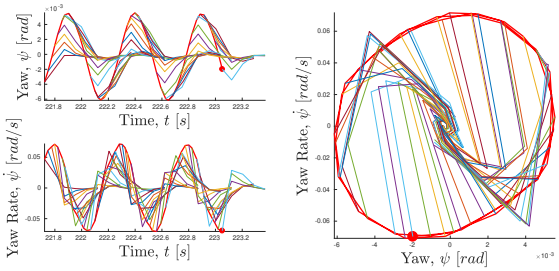
This is especially important in gaits that have very distinct contact modes, such as trotting. In a version of the trot gait, the diagonal pairs of legs are picked up and put down in an alternating pattern. Even with the prediction delay compensation, it is not guaranteed that the predicted phase and the scheduled phase signal the same contact state. Therefore, without this heuristic there would be no force command from the stance legs to keep the robot upright. When the optimized solution contact mode and the scheduled gait sequence return to a synchronized state, the results from the RPC can be confidently used again.

F. Gain Tuning

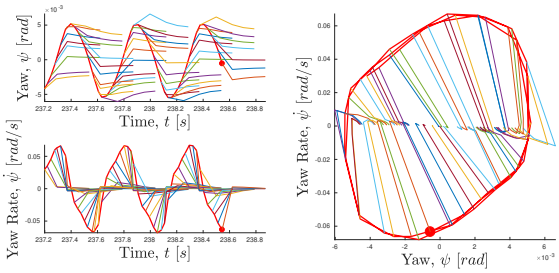
With most controllers, the tuning of the gains is a critical component to its success. To tune the gains, simple desired setpoints were used as we automatically swept over the individual gains for each state. Rough bounds of stability for the gains were found that allowed the robot to automatically tune itself without falling over in simulation by simply finding the gains that produced the best tracking.

Figure 8 shows the two extreme bounds of tuning the yaw behavior of the robot while trotting. While both are stable, they produce very different results. The red lines are the actual measured states of the robot and the multicolored lines are the predicted states returned from the RPC. Low yaw rate gain, $R_{\dot{\psi}}$, produces a more natural and smooth sinusoidal pattern when not forcing the yaw rate to go to zero as is the case with the high gain. The tracking error for the states is used as a metric for tuning, as is the predicted state error. When the predicted states also have good tracking, then the controller is better conditioned and results produce smoother, less sporadic results.

The predicted state trajectory does not line up with the actual state because the prediction model is based on simple uncoupled linear dynamics while the actual robot is a highly nonlinear hybrid system that must adhere to its natural physics regardless of how much the controller fights it. This will be addressed in related work. However, for the purposes of tuning the gains, it is enough to attempt to track as closely to a simple setpoint as possible.



(a) Low Gain, $R_{\dot{\psi}} = 1$



(b) High Gain, $R_{\dot{\psi}} = 500$

Fig. 8: **Intuitive Gain Tuning.** Red lines indicate the actual motion of the robot and the thin multicolored lines represent the predicted states at each moment in time as the yaw rate gain is swept from $R_{\dot{\psi}} = 1$ in (8a) to $R_{\dot{\psi}} = 500$ in (8b).

IV. Results

The RPC was implemented successfully on the MIT Cheetah 3 robot platform. The optimization is solved using the freely available IPOPT NLP solver [21] interfaced in C++. It runs on a separate thread, but on the same ADL embedded Quad core PC with a 2nd Gen Core i7 CPU as the main control loop described in [17]. As mentioned throughout the paper, the RPC runs asynchronously to the main control loop. Therefore while the control loop runs at a fixed $1kHz$, the RPC solution time varies based on computing power available.

We note that the stability of the robot improves as solve frequency increases, with good stability over $40Hz$, decent stability $20-40Hz$, and unstable under $20Hz$. Since generally the heuristic-based initial guess is designed to provide an adequate solution to at least remain stable, we can limit the maximum solve time to be $50ms$ which corresponds to the lowest stable frequency and rely on the heuristics if a solution was not found. In practice the solver generally runs at $60-80Hz$ on the robot during steady state locomotion and around $40-60Hz$ during worst-case large disturbances as the optimization must search further from the heuristic regularization for better solutions.

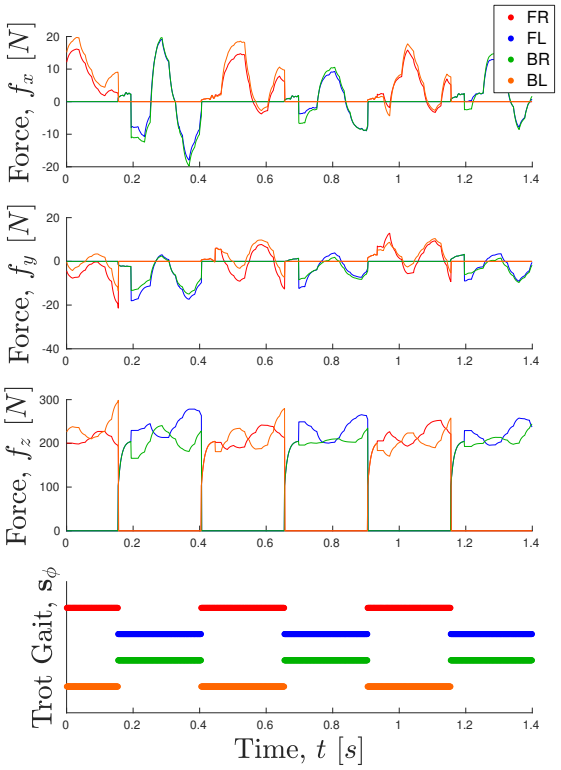
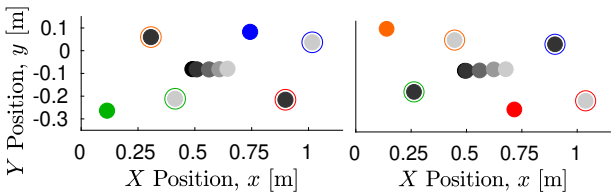


Fig. 9: **Ground Reaction Forces.** Optimized ground reaction forces during a trotting gait. Forces are smoothly filtered even though the RPC returns discrete constant forces at a lower frequency than the control loop.

The xyz ground reaction forces resulting from the RPC are shown in Figure 9. The gait schedule contact sequence for the trotting gait is also depicted for the six footsteps. Each diagonal pair of legs is capable of stabilizing the robot while trotting in place. A discrete jump in the forces can be seen at the beginning of each new contact mode, signifying the switch from the short period of heuristic force calculation to the RPC solution.

In addition to selecting the ground reaction forces at the contact points, future footsteps are also chosen. The benefit of simultaneously optimizing for both is that footsteps are chosen in the same context as future stabilizing forces. Figure 10 shows two example predicted footstep optimization results during $1.4\frac{m}{s}$ forward locomotion using the robot with a trotting gait. The figure depicts a 2D projection in the XY plane of the (x, y) coordinates for the CoM, predicted CoM, contact feet, and predicted footstep locations. In Figure 10a, the front left (blue) and back right (green) feet are on the ground while the next footstep locations for the front right and back left feet are solved for. The prediction horizon is long enough such that the next footsteps for the front left and back right feet are also already being solved for before they have entered their swing phase.

The robot was able to closely track the desired velocities during the experiment both forwards and backwards. The body's orientation throughout is modulated close to level



(a) Timestep t_1 (b) Timestep t_2

Fig. 10: **Predicted Footstep Locations.** Solid colored dots represent the current contact foot locations while the colored circles signify the predicted footstep locations. The solid circle inside of the predicted footsteps corresponds to the future time segment of touchdown, where the darker the circle, the closer to the current time. CoM predicted trajectory is shown in the middle with decreasingly dark dots over time.

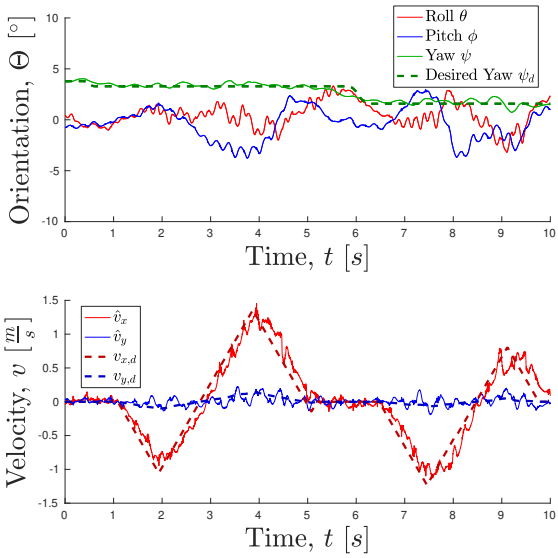


Fig. 11: **Velocity Tracking.** The robot was given forward and backwards velocity commands in the x direction and was able to closely track the desired values while keeping its body orientation close to flat.

with minor fluctuations, never exceeding more than 5° which usually occurs during the sharp velocity changes. However, it returns flat when the robot is in steady state. Figure 11 shows the robot's ability to track commanded translational velocities. The maximum commanded velocity is $1.4\frac{m}{s}$ during trotting. Turning rate commands were also sent to the robot as seen in Figure 12. Maximum turning rate given was $2\frac{rad}{s}$ with a mean orientation error in roll of 0.47° and 0.48° in pitch throughout.

V. Conclusion

This paper showed a real-time implementation of a nonlinear optimization based Regularized Predictive Controller on the MIT Cheetah 3 quadruped robot. While optimization-based controllers for legged robots have been explored in many related works, it is common for the performance to be affected by non-convergence and computation time problems. The real contribution of this paper is to describe the

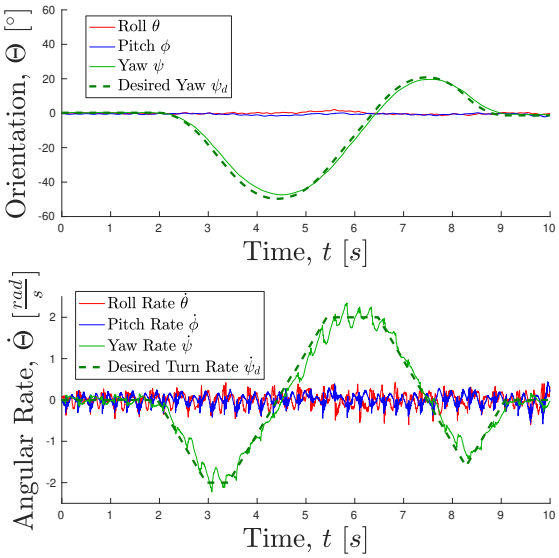


Fig. 12: **Turn Rate Tracking.** The robot was given a turning rate command which it was able to track with little error over the duration of the test.

implementation details that mitigate some of these problems that are typical of nonlinear optimization controllers. With the presented work, we were able to show results of the controller successfully controlling the robot through dynamic locomotion in real-time.

Using the custom realistic dynamics simulation environment on a more powerful desktop processor, the optimization ran at approximately $120 - 150Hz$, sometimes reaching up to $200Hz$. With a more powerful CPU in the robot, we could exceed this solution frequency and improve performance while relying less on some of the compensation techniques described throughout this paper. However, with the current hardware setup, the robot still performs well.

Future work will continue to develop the controller and heuristics to improve both the optimization solution time and the stable operating regions. Work is also underway to integrate perception with the controllers. The controller should be able to operate in most situations blindly, while being able to improve performance through the integration of perception information.

VI. Acknowledgments

The authors would like to thank the members of the MIT Biomimetic Robotics Lab, in particular Ben Katz, Jared Di Carlo, and Quan Nguyen for assistance during experiments, as well as hardware platform support.

References

[1] S. Kuindersma, R. Deits, M. Fallon, A. Valenzuela, H. Dai, F. Permenter, T. Koolen, P. Marion, and R. Tedrake, "Optimization-based locomotion planning, estimation, and control design for the atlas humanoid robot," *Autonomous Robots*, vol. 40, pp. 429–455, Mar 2016.

[2] 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, pp. 259–272, Jan 2017.

[3] M. Focchi, R. Orsolino, M. Camurri, V. Barasuol, C. Mastalli, D. G. Caldwell, and C. Semini, "Heuristic planning for rough terrain locomotion in presence of external disturbances and variable perception quality," *CoRR*, vol. abs/1805.10238, 2018.

[4] C. Dario Bellicoso, F. Jenelten, P. Fankhauser, C. Gehring, J. Hwangbo, and M. Hutter, "Dynamic locomotion and whole-body control for quadrupedal robots," in *2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pp. 3359–3365, Sep. 2017.

[5] C. D. Bellicoso, F. Jenelten, C. Gehring, and M. Hutter, "Dynamic locomotion through online nonlinear motion optimization for quadrupedal robots," no. 99, 2018-01-17.

[6] P.-B. Wieber, "Trajectory free linear model predictive control for stable walking in the presence of strong perturbations," in *Humanoids*, 2006.

[7] D. Dimitrov, A. Sherikov, and P.-B. Wieber, "A sparse model predictive control formulation for walking motion generation," in *Intelligent Robots and Systems (IROS), 2011 IEEE/RSJ International Conference on*, pp. 2292 –2299, sept. 2011.

[8] T. Apgar, P. Clary, K. Green, A. Fern, and J. W. Hurst, "Fast online trajectory optimization for the bipedal robot cassie," in *Robotics: Science and Systems*, 2018.

[9] 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 *Intelligent Robots and Systems (IROS), 2015 IEEE/RSJ International Conference on*, pp. 3346–3351, Sept 2015.

[10] A. W. Winkler, F. Farshidian, M. Neunert, D. Pardo, and J. Buchli, "Online walking motion and foothold optimization for quadruped locomotion," *2017 IEEE International Conference on Robotics and Automation (ICRA)*, pp. 5308–5313, 2017.

[11] A. W. Winkler, F. Farshidian, D. Pardo, M. Neunert, and J. Buchli, "Fast trajectory optimization for legged robots using vertex-based ZMP constraints," *CoRR*, vol. abs/1705.10313, 2017.

[12] M. Neunert, C. de Crousaz, F. Furrer, M. Kamel, F. Farshidian, R. Siegwart, and J. Buchli, "Fast nonlinear model predictive control for unified trajectory optimization and tracking," in *IEEE Int. Conf. on Rob. and Automation*, pp. 1398–1404, May 2016.

[13] M. Kamel, K. Alexis, M. Achtelik, and R. Siegwart, "Fast nonlinear model predictive control for multicopter attitude tracking on so(3)," in *2015 IEEE Conference on Control Applications (CCA)*, pp. 1160–1166, Sep. 2015.

[14] R. Wittmann, A. Hildebrandt, D. Wahrmann, D. Rixen, and T. Buschmann, "Real-time nonlinear model predictive footstep optimization for biped robots," in *2015 IEEE-RAS 15th International Conference on Humanoid Robots (Humanoids)*, pp. 711–717, Nov 2015.

[15] F. Farshidian, E. Jelavic, A. Satapathy, M. Giftthaler, and J. Buchli, "Real-time motion planning of legged robots: A model predictive control approach," *CoRR*, vol. abs/1710.04029, 2017.

[16] 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," *CoRR*, vol. abs/1712.02889, 2017.

[17] G. Bledt, M. J. Powell, B. Katz, J. D. Carlo, P. M. Wensing, and S. Kim, "MIT cheetah 3: Design and control of a robust, dynamic quadruped robot," in *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems*, (Madrid, Spain), Oct. 2018.

[18] J. D. Carlo, P. M. Wensing, B. Katz, G. Bledt, and S. Kim, "Dynamic locomotion in the MIT cheetah 3 through convex model-predictive control," in *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems*, (Madrid, Spain), Oct. 2018.

[19] G. Bledt, P. M. Wensing, and S. Kim, "Policy-regularized model predictive control to stabilize diverse gaits for the MIT cheetah," in *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems*, (Vancouver, Canada), Sept. 2017.

[20] M. Diehl, H. Bock, H. Diedam, and P.-B. Wieber, "Fast direct multiple shooting algorithms for optimal robot control," in *Fast Motions in Biomechanics and Robotics* (M. Diehl and K. Mombaur, eds.), vol. 340 of *Lecture Notes in Control and Information Sciences*, pp. 65–93, Springer Berlin Heidelberg, 2006.

[21] A. Wächter and L. T. Biegler, "On the implementation of an interior-point filter line-search algorithm for large-scale nonlinear programming," *Math. Program.*, vol. 106, pp. 25–57, Mar. 2006.