Robotics: Science and Systems 2011
Los Angeles, CA, USA, June 27-30, 2011

Infinite-Horizon Model Predictive Control for Periodic Tasks with Contacts

Tom Erez*,Yuval Tassa†, and Emanuel Todorov‡
*Computer Science and Engineering
Washington University in St. Louis, Missouri
etom@cse.wustl.edu
†Interdisciplinary Center for Neural Computation
Hebrew University, Jerusalem, Israel
tassa@alice.nc.huji.ac.il
‡Applied Mathematics and Computer Science & Engineering
University of Washington, Seattle, USA
todorov@cs.washington.edu

***Abstract*—We present a method that combines offline trajectory optimization and online Model Predictive Control (MPC), generating robust controllers for complex periodic behavior in domains with unilateral constraints (e.g., contact with the environment). MPC offers robust and adaptive control even in high-dimensional domains; however, the online optimization gets stuck in local minima when the domains has discontinuous dynamics. Some methods of trajectory optimization that are immune to such problems, but these are often too slow to be applied online.**

In this paper, we use offline optimization to find the limit-cycle solution of an infinite-horizon average-cost optimal-control task. We then compute a local quadratic approximation of the Value function around this limit cycle. Finally, we use this quadratic approximation as the *terminal cost* of an online MPC.

This combination of an offline solution of the infinite-horizon problem with an online MPC controller is known as Infinite Horizon Model Predictive Control (IHMPC), and has previously been applied only to simple stabilization objectives. Here we extend IHMPC to tackle periodic tasks, and demonstrate the power of our approach by synthesizing *hopping* behavior in a simulated robot. IHMPC involves a limited computational load, and can be executed online on a standard laptop computer. The resulting behavior is extremely robust, allowing the hopper to recover from virtually any perturbation.

In real robotic domains, modeling errors are inevitable. We show how IHMPC is robust to modeling errors by altering the morphology of the robot; the same controller remains effective, even when the underlying infinite-horizon solution is no longer accurate.

I. Introduction

Methods of optimal control derive control policies from cost functions that penalize undesirable states. This is an appealing paradigm, because it offers the system designer an intuitive scheme — it is easier to specify which states are desirable than to directly craft the policy that realizes these goals.

The *finite-horizon* criterion seeks to minimize the future cumulative cost over some predefined planning horizon. The repeated online solution of the finite-horizon problem for an ever-receding horizon is called Model Predictive Control (MPC). Online optimization is possible because this class of problems is relatively easy to solve, but may result in undesirable "myopic" behavior due to the limited planning horizon. In particular, it is ineffective in domains with contacts (section IV).

The *infinite-horizon* criterion poses the optimization problem in terms of the average cost over an infinitely-distant horizon. This formulation is used in domains where there are well-defined terminal states, as well as domains where the solution is periodic and forms a limit cycle (e.g., gait synthesis and optimization). This class of optimization problems is usually too computationally-intensive to be solved online.

The strengths of these two formulations can be combined by using the cost-to-go of an infinite-horizon solution as the terminal cost of the finite-horizon problem. This scheme is called Infinite-Horizon MPC (IHMPC).

Existing IHMPC algorithms can only tackle domains with fixed goal states (section II). In contrast, this paper tackles the more general problem where the optimal behavior results in a limit cycle. We use an offline optimization scheme to construct a locally-quadratic approximation of the infinite-horizon cost-to-go around the optimal limit-cycle. We then use this approximation as a terminal cost for online IHMPC.

We show how this approach can generate robust control for a simulated hopping robot (figure I) in real time. The optimal limit cycle is found through offline optimization (section III-B), and the infinite-horizon average-cost value function (section VI) is fitted around the closed trajectory; this approximation is used for IHMPC (section VII-B). The computation of the MPC optimization is fast enough to allow real-time simulation and control on a standard laptop computer. The resulting controller yields robust behavior that can effectively recover from any perturbation, as illustrated by a movie demonstrating our results (which is available at goo.gl/ymzUY).

Finally, we address the question of modeling errors. The methods presented here use model-based dynamic trajectory optimization both online and offline; however, such dynamic models would always be somewhat inaccurate for real robots. Robustness with respect to modeling errors is studied by

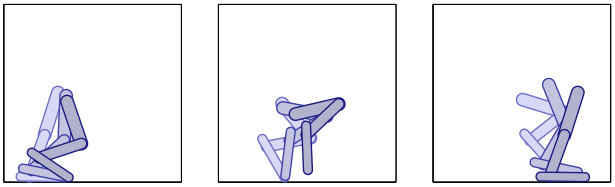

Fig. 1. The hopping robot's limit cycle.

altering the dynamics of the simulated plant, so that the optimization uses an incongruent dynamic model (section VII-C). Even in this case, our controller can generate the desired behavior (figure 5) and recover from perturbations.

II. Related work

Local algorithms of optimal control fall into two broad categories [8], according to the representation of the search space: *simultaneous* methods explicitly represent the trajectory in state space, treating the dynamics as constraints; in contrast, *sequential* methods represent only the control sequence, and use numerical integration to evaluate the resulting trajectory. The sequential approach often employs less variables, and the resulting trajectories are guaranteed to be dynamically consistent. The simultaneous approach allows the optimization to consider infeasible trajectories; this may prevent getting stuck in some local minima, but requires machinery for enforcing dynamical consistency.

The basic algorithm for sequential trajectory optimization in a nonlinear system is Pontryiagin's minimum principle [19], a generalization of the Euler-Lagrange equations that provides a first-order criterion for identifying a locally-optimal trajectory. More elaborate proposals include algorithms by Jacobson and Mayne [12] (see appendix), Bryson and Ho [6] and Bertsekas and Dunn [9].

MPC has been studied extensively (see reviews by Morari and Lee [16], Bertsekas [4], Mayne et al. [14], Diehl et al. [8], and references therein), but its application to robotic domains is only starting to gain popularity [1]. Chen and Allgöwer [7] seem to be the first to suggest the combination of MPC with an infinite-horizon optimization problem, and this approach has been studied extensively in the past decade [18, 2, 11]. However, current IHMPC algorithms make the strong assumption that the task is specified in terms of reaching a goal state. In such a case, only a finite amount of time is spent away from this target, and so the conditions around the goal state dominate the infinite-horizon considerations. Even if the domain exhibits non-linear dynamics, a linear approximation can be used effectively in some small region around the goal state. Together with a quadratic approximation of the cost function, the infinite-horizon problem takes the familiar form of a Linear-Quadratic Regulator (LQR), and can be solved using Ricatti equations.

The optimization criterion of infinite-horizon average-cost has been studied in the past two decades [20, 10, 13, 21]; it was used for policy search [3], and successfully applied to gait optimization [23]. Local methods of optimization that use a simultaneous representation include multiple shooting [5] and space-time constraints [26], and this approach has been applied to gait design [25, 15]. Popović and Wu [27] presented a method where offline optimization (through Evolutionary Computation) was complemented by Quadratic Programming during runtime to generate robust locomotion behavior from motion-capture data.

In contrast to existing IHMPC algorithms, the method proposed in this paper allows us to tackle domains with no single goal state. While local feedback controllers can be fit around optimal trajectories found by offline optimization, MPC offers much wider basins of attraction, as the online re-optimization generates robust control that can recover the desired behavior from any perturbation.

III. Background

We consider systems with non-linear, discrete-time dynamics of the form:

$$\mathbf{x}(i+1) = \mathbf{f}(\mathbf{x}(i), \mathbf{u}(i)) \tag{1}$$

where $\mathbf{x} \in \mathbb{R}^n$ is the system's state, and $\mathbf{u} \in \mathbb{R}^m$ is the control signal. The optimization criterion is specified by the cost function $\ell(\mathbf{x}, \mathbf{u})$.

A. Trajectory optimization and MPC

Given a planning horizon N, we seek the open-loop control sequence $\mathbf{U} = \{\mathbf{u}(1), \ldots, \mathbf{u}(N-1)\}$ that minimizes the cumulative cost:

$$V\big(\mathbf{x}(1), 1\big) = \min_{\mathbf{U}} \sum_{i=1}^{N-1} \ell\big(\mathbf{x}(i), \mathbf{u}(i)\big) + \ell_N\big(\mathbf{x}(N)\big) \tag{2}$$

where $\mathbf{x}(i)$ is defined by (1) for $i > 1$, and ℓ_N is an optional state-dependent cost function which is applied to the terminal state of the trajectory.

Bellman's equation for this case yields a *time-dependent* value function, defined recursively as the optimal cost-to-go:

$$V(\mathbf{x}, i) = \min_{\mathbf{u}} \left[\ell(\mathbf{x}, \mathbf{u}) + V\big(\mathbf{f}(\mathbf{x}, \mathbf{u}), i+1\big)\right]. \tag{3}$$

Since the value function is time-dependent, the effective planning horizon of states at the latter part (tail end) of the trajectory is very short, which may cause myopic behavior in these states. The terminal cost function ℓ_N mitigates that problem, since it can effectively inform the controller about all the events which lie beyond its planning horizon. However, crafting a terminal cost function that yields the intended result can be hard.

Finite-horizon optimization problems can be solved by both simultaneous and sequential approaches; we focus here on the sequential approach. Algorithms in this category (see section II) share a common structure: given a fixed first state, and initialized with some control sequence, every iteration seeks an improved control sequence. The basic structure of a single iteration consists of two phases: first, a new nominal trajectory is found by integrating the current control sequence forward

in time using equation (1). Then, the trajectory is swept backwards, and a modification to the control sequence is computed. These two passes are repeated until no improvement is found.

The computation of the modified control sequence often involves the integration of the time-dependent value function along the nominal trajectory. In such cases, every step of the backwards pass computes a local approximation of the value function around a certain state by integrating backwards the local approximation around the next state, following equation (3). Often, a locally-quadratic model of the value function is used, parametrized by a linear component $V_{\mathbf{x}}(i)$ and a quadratic component $V_{\mathbf{xx}}(i)$. In such cases the backwards step has the form:

$$\left\{V_{\mathbf{x}}(i), V_{\mathbf{xx}}(i)\right\} = B\big(\mathbf{x}(i), V_{\mathbf{x}}(i+1), V_{\mathbf{xx}}(i+1)\big). \quad (4)$$

We include an explicit description for such an algorithm in the appendix, where the backwards step B is computed by equations (11)-(13). These algorithms are fast enough to be used for planning in Model-Predictive Control (MPC).

B. Limit-Cycle Optimization

In this case, we seek a policy $\mathbf{u} = \pi(\mathbf{x})$ that minimizes the average cost along the trajectory in the limit of infinitely-long horizon:

$$c = \min_{\pi} \lim_{\mathcal{N}\to\infty} \frac{1}{\mathcal{N}} \sum_{i=1}^{\mathcal{N}} \ell\Big(\mathbf{x}(i), \pi\big(\mathbf{x}(i)\big)\Big). \quad (5)$$

Given the optimal policy, the value function of the infinite-horizon problem is defined as the deviation of the future cumulative cost from the average:

$$\tilde{V}\big(\mathbf{x}(1)\big) = \lim_{\mathcal{N}\to\infty} \left[\sum_{i=1}^{\mathcal{N}} \ell\Big(\mathbf{x}(i), \pi\big(\mathbf{x}(i)\big)\Big) - \mathcal{N}c\right]. \quad (6)$$

Bellman's equation for this formulation is simpler, as it is no longer time dependent:

$$c + \tilde{V}(\mathbf{x}) = \min_{\mathbf{u}} \Big[\ell(\mathbf{x}, \mathbf{u}) + \tilde{V}\big(\mathbf{f}(\mathbf{x}, \mathbf{u})\big)\Big] \quad (7)$$

A sequential approach is inapplicable to solve infinite-horizon problems. Instead, we apply a simultaneous approach to the optimization of limit cycles, which takes the general form:

$$\min_{\mathbf{X},\mathbf{U}} \sum_{i=1}^{N} \ell\big(\mathbf{x}(i), \mathbf{u}(i)\big) \quad \text{s.t. } \forall i : \psi\big(\mathbf{x}(i), \mathbf{u}(i), \mathbf{x}(i+1)\big) = 0$$

where N is the period, $\mathbf{X} = \{\mathbf{x}(i)\}_{i=1}^{N}$ is a sequence of states, $\mathbf{U} = \{\mathbf{u}(i)\}_{i=1}^{N}$ is a sequence of controls, and the function

$$\psi(\mathbf{x}, \mathbf{u}, \mathbf{x}') = \mathbf{x}' - \mathbf{f}(\mathbf{x}, \mathbf{u})$$

imposes dynamical consistency between every consecutive pair of states, treating equation (1) as a constraint. Since we focus on limit cycles, we define $N + 1 \equiv 1$ so that the last state $\mathbf{x}(N)$ is followed by the first state $\mathbf{x}(1)$.

The optimization algorithm searches in the space of all rings $\mathbf{X}$ for a limit cycle that minimizes the cumulative cost. In order to reduce the dimensionality of the search space, we build a model of the plant that solves for *inverse dynamics*: given a pair of states $\mathbf{x}, \mathbf{x}'$, we compute the control signal $\mathbf{u}$ that minimizes the dynamical error $||\psi(\mathbf{x}, \mathbf{u}, \mathbf{x}')||^2$. This allows us to maintain only a representation of the state-space trajectory $\mathbf{X}$, and have it implicitly define $\mathbf{U}$.

In practice, we can handle the constraints ψ by using a penalty method, yielding the unconstrained optimization problem

$$\min_{\mathbf{X}} \sum_{i=1}^{N} \Big[\ell\big(\mathbf{x}(i), \mathbf{u}(i)\big) + \lambda \big|\big|\psi\big(\mathbf{x}(i), \mathbf{u}(i), \mathbf{x}(i+1)\big)\big|\big|^2\Big]$$

which can be solved using standard optimization methods.

IV. Why MPC is not enough

The method of MPC is designed to avoid the problem of myopic behavior in finite-horizon planning: while an entire trajectory is planned, only the first action is executed, and planning starts again from the resulting new state. In order to be applicable for robotic domains, the process of re-planning must be very rapid. This is achieved by warm-starting the optimization with the entire optimized trajectory from the previous iteration, with a single time-shift. The effectiveness of MPC rests on the assumption that the solution at the previous timestep is similar to the solution at this timestep, and therefore the warm-start will allow for efficient optimization.

This assumption is disrupted if the tail of the trajectory falls into a local optimum. This will have no immediate effect, because only the first action is actually executed by the MPC, but as the planning horizon recedes, more and more of the trajectory is pushed into that local minimum. If at some stage the local optimality vanishes, the trajectory used to initialize the re-planning is no longer almost-optimal. This usually leads to the failure of MPC, since there is not enough time for re-optimization.

In section VII-B, we describe a one-legged hopping robot whose task is to maintain a fixed horizontal velocity. Intuitively, the optimal behavior we expect in such a domain is a hopping gait. However, applying regular MPC to this domain results in catastrophic failure. To understand why, consider a trajectory where the last few states could involve ground collision. Such a collision would necessarily slow down the hopper, which would impede the performance of the task in the short term. As such, in the absence of an adequately long planning horizon, the optimal solution is myopic, causing the hopper to retract its leg; by avoiding the ground, the hopper tries to maintain its air velocity just a few timesteps longer. However, as the planning horizon of the MPC recedes, the locally-optimal avoidance maneouver becomes more complicated, and eventually ground impact becomes inevitable. At that stage, the MPC must plan the foot landing, but the optimization is initialized with a suboptimal trajectory that involves a bizarre contortion towards the end. Unwinding this suboptimal behavior is probably impossible within the time constraints of the robotic application, leading to the failure of MPC.

V. Infinite-horizon model predictive control

In IHMPC, we compute the infinite-horizon value $\tilde{V}(\mathbf{x})$ of a given state $\mathbf{x}$ by breaking the infinite sum in (6) into two parts:

$$\lim_{\mathcal{N}\to\infty}\left[\sum_{i=1}^{\mathcal{N}}\ell\Big(\mathbf{x}(i),\pi\big(\mathbf{x}(i)\big)\Big)-\mathcal{N}c\right]= \sum_{i=1}^{N-1}\ell\Big(\mathbf{x}(i),\pi\big(\mathbf{x}(i)\big)\Big)-(N-1)c+ \lim_{\mathcal{N}\to\infty}\left[\sum_{i=N}^{\mathcal{N}}\ell\Big(\mathbf{x}(i),\pi\big(\mathbf{x}(i)\big)\Big)-\mathcal{N}c\right]. \quad (8)$$

Note that the first RHS sum is the finite-horizon value function (2), and the latter sum is simply $\tilde{V}\big(\mathbf{x}(N)\big)$. Therefore:

$$\tilde{V}\big(\mathbf{x}(1)\big)=V\big(\mathbf{x}(1)\big)+\tilde{V}\big(\mathbf{x}(N)\big)+\text{constant}.$$

This identity implies a compositionality of controllers: if we use the infinite horizon value function $\tilde{V}$ as the terminal cost function ℓ_N in (2), the MPC solution would be identical to the globally-optimal controller derived directly from $\tilde{V}$.

However, $\tilde{V}$ cannot be computed online, and in high-dimensional domains it is impossible to pre-compute $\tilde{V}(\mathbf{x})$ for the entire state space. Previous studies of IHMPC focused on tracking tasks where a well-defined goal state is available. In this case, LQR/LQG theory can be used to find $\tilde{V}$ for the goal state. However, in some domains of autonomous behavior the cost function does not specify a specific target state. In this case, a different approach is needed to approximate $\tilde{V}$. We use offline trajectory optimization to identify the optimal limit cycle, and construct a local quadratic approximation of $\tilde{V}$ around it (section VI). For a given $\mathbf{x}(1)$, we can find a good approximation of $\tilde{V}\big(\mathbf{x}(1)\big)$ using only finite horizon optimization (which can be computationally very efficient), as long as MPC can find a solution where $\mathbf{x}(N)$ is close to that limit cycle.

VI. Approximating the infinite horizon value function

Given an optimized closed trajectory $\mathbf{X}$ and the corresponding open-loop sequence $\mathbf{U}$, we approximate the value function of the infinite-horizon problem locally as a quadratic function. This means that for each of the points $\mathbf{x}(i)\in\mathbf{X}$ we seek the coefficients $\tilde{v}(i),\tilde{V}_{\mathbf{x}}(i),\tilde{V}_{\mathbf{xx}}(i)$ so that for small $\delta\mathbf{x}$,

$$\tilde{V}(\mathbf{x}(i)+\delta\mathbf{x};i)\approx\tilde{v}(i)+\delta\mathbf{x}^{\mathsf{T}}\tilde{V}_{\mathbf{x}}(i)+\frac{1}{2}\delta\mathbf{x}^{\mathsf{T}}\tilde{V}_{\mathbf{xx}}(i)\delta\mathbf{x}.$$

The constant terms $\tilde{v}(i)$ can be computed directly, while the coefficients $\tilde{V}_{\mathbf{x}}(i),\tilde{V}_{\mathbf{xx}}(i)$ can be estimated using least-squares.

A. The Constant Terms $\tilde{v}$

Since we can measure the cost ℓ along every point of the limit cycle, we can calculate the average cost per step

$$c=\frac{1}{N}\sum_{i=1}^{N}\ell\big(\mathbf{x}(i),\mathbf{u}(i)\big).$$

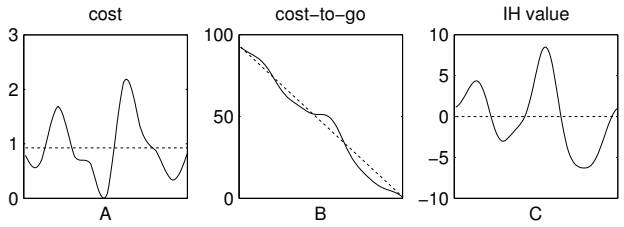

Fig. 2. An illustration of the calculation of the infinite-horizon value function along a limit cycle (described in section VI-A). **A.** the cost ℓ at every point along a limit cycle (solid), and the average cost (dashed). **B.** the finite horizon cost-to-go (solid) and the average cost-to-go (dashed), integrated along one cycle of the closed trajectory. **C.** the resulting infinite-horizon average-cost value function, shifted so that its mean over the entire limit cycle is zero.

We define the cost-to-go along the limit cycle

$$V(i)=\sum_{k=i}^{N}\ell\big(\mathbf{x}(i),\mathbf{u}(i)\big)$$

and the average cost-to-go

$$\bar{V}(i)=(N-i)c.$$

The infinite-horizon value function is the deviation of the cost-to-go from the average cost-to-go

$$\tilde{V}(i)=V(i)-\bar{V}(i)+b$$

(see figure 2), shifted by the scalar b to enforce

$$\sum_{i=1}^{N}\tilde{V}(i)=0.$$

This computation is illustrated in figure 2.

B. Estimating the Quadratic Model

Equation (4) describes a relationship between two consecutive quadratic models of the time-dependent value function (3). Equation (7) describes a very similar relationship between two parts of the same value function around two temporally-consecutive states. However, although equations (3) and (7) differ only in the constant term, a straightforward backward integration of equation (4) along the limit cycle cannot always be used to evaluate the value function of the infinite-horizon problem. The backward integration used in the finite-horizon case relies on the dynamic-programming principle that the past can affect the future, but not vice-versa. This principle does not hold in limit cycles, where every state is visited both before and after every other state. We found that in practice, simple backward integration of equation (4) along several rounds of the limit cycle can be effective for some domains (section VII-A), but fails to produce robust behavior in others (section VII-B).

As an alternative, we can pose equation 4 in terms of least-squares minimization. Given some values of a quadratic model $\tilde{V}_{\mathbf{x}}(i),\tilde{V}_{\mathbf{xx}}(i)$ and its successor $\tilde{V}_{\mathbf{x}}(i),\tilde{V}_{\mathbf{xx}}(i)$, we can define the backward step mismatch as

$$\Psi(i)=\{\tilde{V}_{\mathbf{x}}(i),\tilde{V}_{\mathbf{xx}}(i)\}-B\big(\tilde{V}_{\mathbf{x}}(i+1),\tilde{V}_{\mathbf{xx}}(i+1)\big).$$

The difference between two quadratic models can be defined in several ways; here we take the simple approach of calculating the difference between corresponding coefficients. We seek a sequence of quadratic models that minimize the backward step mismatch:

$$\min_{\{\tilde{V}_{\mathbf{x}}(i),\tilde{V}_{\mathbf{xx}}(i)\}_{i=1}^{N}} ||\Psi(i)||^2 .$$

This is a nonlinear sum-of-squares optimization problem whose variables are the coefficients of the quadratic models around the ring. Although the number of variables in this problem is quadratic in the dimensionality of the domain, note that only variables of consecutive pairs interact. Therefore, the Hessian of this optimization problem is sparse, and it can be efficiently solved using standard optimization algorithms.

VII. Results

We present results from two domains. First, we tackle a 2D domain. This problem is small enough to afford a global solution through state-space discretization, which serves as ground truth. This allows us to evaluate the quality of our approximations. Then, we apply our method to a simulated domain of a one-legged hopping robot.

A. 2D Problem

A variant of this non-linear problem was first proposed by Todorov [24], and we adopt it as a benchmark because the optimal solution was shown to be a limit cycle. First, we obtain ground-truth for the optimal policy by discretizing the state space with a 200x200 grid, and solve the resulting 40,000-states Markov Decision Process which corresponds to the infinite-horizon problem. The resulting policy is shown at the top left of figure 3.

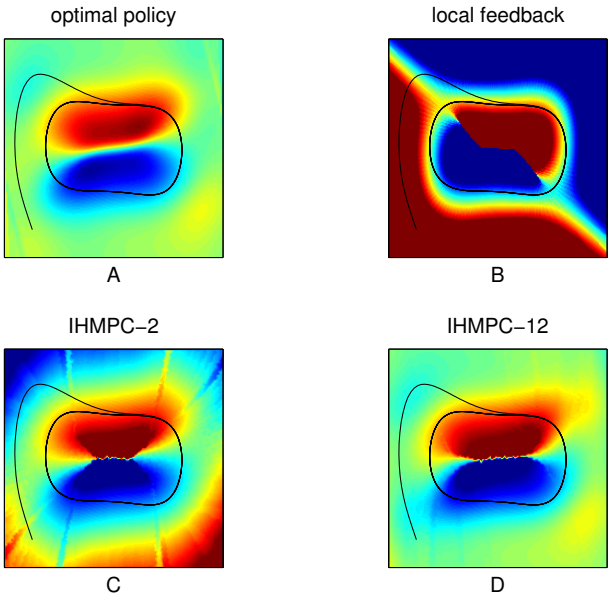

Fig. 3. **A.** the ground-truth policy solved through MDP discretization. **B.** the locally-linear feedback policy computed around the limit cycle. **C.** IHMPC with a lookahead horizon of 2 steps. **D.** IHMPC with a horizon of 12 steps. The complete limit cycle is 150 steps long.

body	length (cm)	radius (cm)	mass (kg)
foot	50	7	8.77
lower leg	50	6	6.33
upper leg	50	5	4.32

TABLE I
Morphological specification of the hopping robot

We then found the optimal limit cycle using the algorithm in section III-B, and confirmed that it matches the limit cycle found by the MDP. We identified quadratic approximations of the value function around every element of the limit cycle according to the algorithm presented in section VI. The locally-linear feedback constructed directly from this quadratic approximation is shown on the top right of figure 3.

Finally, we used IHMPC to compute the policy over all states in the same 200x200 grid. The two lower panels of figure 3 shows the resulting policies for different lengths of the MDP horizon. With a planning horizon of two steps, numerical artifacts are still apparent (bottom left). However, with a planning horizon of 12 steps, the IHMPC policy becomes effectively equal to the true policy almost everywhere.

B. Planar Hopping Robot

This mechanical system of the hopping robot is composed of three body segments and two joints, and so this domain has 10-dimensional state space and two-dimensional control space. The masses and segment lengths are specified in table I. The ground interaction forces are computed using stochastic LCP [22] with $\sigma = 1$. The task requires the hopper's center of mass to maintain a fixed horizontal velocity $\dot{y}_{COM}$ of 1 m/s, while keeping its vertical position x_{COM} around 1 m:

$$\ell(\mathbf{x}, \mathbf{u}) = 10(\dot{y}_{COM} - 1)^2 + 0.1(x_{COM} - 1)^2 + 0.01\,||\mathbf{u}||^2$$

We started by finding an optimal limit cycle with $N = 40$ steps of 20msec. The optimization of the hopping gait was implemented in MATLAB, and took about an hour of computation on a standard dual-core T9300 Intel processor. Fitting the quadratic approximation of the value function around the limit cycle took about 5 minutes of computation. Both these stages are run offline only once.

We then applied IHMPC with a planning horizon of 10 steps. Every MPC optimization loop was allowed at most 50msec or 5 iterations. When the simulation is unperturbed, the MPC optimization converged in a single iteration in every timestep. This allowed us to apply IHMPC in real-time. IHMPC was able to generate robust behavior over the entire state-space. The hopping behavior is best illustrated by a movie depicting the hooper in action, which is available online at `goo.gl/ymzUY`. The basin of attraction of the IHMPC effectively covers the entire volume of state space, and the hopping robot can recover from any perturbation and resume its gait. Even when the hopper is thrown to the ground (frames

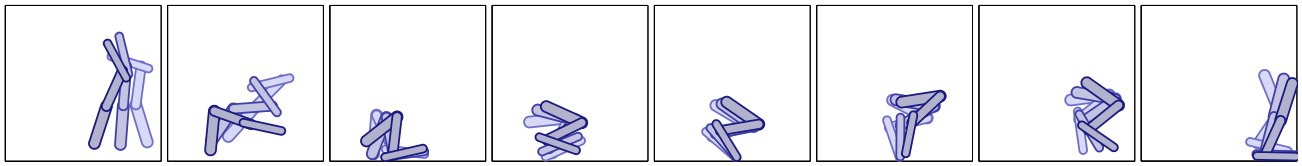

Fig. 4. IHMPC recovering from a starting state that is far from the limit cycle. Note how in the first frame on the right, the hopping robot is flying upside-down, and yet manages to eventually get a foothold and push itself back up.

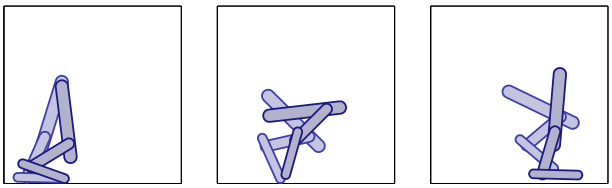

Fig. 5. The limit cycle of a hopping robot with altered morphology (the top body's length is extended by 60% to 80 cm). IHMPC can recover stable hopping even as the planner still uses the original model.

1 and 2 in figure 4), it can find an appropriate motor sequence to get up (frames 3 and 4) and resume its hopping (frames 5-8).

C. Robustness to modeling errors

We use model-based optimization for both the offline and the online phases of our IHMPC algorithm. However, it is impossible to build a perfectly-accurate dynamical model of a physical robot. Therefore, in order to be useful for real-world robotic applications, any model-based control method must be robust to modeling errors. We examined this question by modifying the model used for plant simulation, making it incongruent with the model used for optimization. We increased the length of the top segment by 60%, from 50cm to 80cm. In this case, the original limit cycle is no longer the optimal gait for the modified morphology, and the MPC optimizations are using the wrong model of the dynamics. However, IHMPC was able to maintain effective hopping in this case as well (figure 5).

VIII. Conclusion

The main contribution of this paper is the presentation of an algorithm for Infinite Horizon Model Predictive Control (IHMPC) for tasks with no specified goal state. We show that IHMPC can generate robust behavior in domains with discontinuities and unilateral constraints, even at the face of extreme external perturbations. We also show how our controller can maintain the desired behavior in the face of significant modeling errors.

Our results show a hopping robot that can get up when tossed to the ground. The controller's capacity to figure out how to get up is not part of the behavioral repertoire that was pre-computed offline. Instead, it is a result of MPC online trajectory optimization. This highlights how IHMPC allows for "motor creativity", as the online optimization can tackle any perturbation, as long as it can find a finite trajectory that terminates close enough to the optimal limit cycle. This kind of open-ended control is essential for robots who share the space with unpredictable humans and their ever-shifting environment.

Appendix

In the experiments described in section VII we used Differential Dynamic Programming (DDP), an algorithm by Jacobson and Mayne [12] which uses second-order expansion of both dynamics and cost to iteratively find improved trajectories given an initial state. Here we only repeat the main equations (as presented by [17]), and refer the interested reader to [12] for further details.

Every iteration involves a *backward pass* along the current $(\mathbf{x}, \mathbf{u}, i)$ trajectory, recursively constructing a quadratic approximation to $V(\mathbf{x}, i)$, followed by a *forward pass* which applies the new control sequence to form a new trajectory.

The backward pass integrates a quadratic approximation of the value function backward in time along the trajectory. Given a quadratic model around $\mathbf{x}(i+1)$, we compute the quadratic model $V_{\mathbf{x}}(i), V_{\mathbf{xx}}(i)$ around $\mathbf{x}(i)$ as a function of $V_{\mathbf{x}}(i+1), V_{\mathbf{xx}}(i+1)$ and the quadratic models of $\mathbf{f}, \ell$ around $\mathbf{x}(i)$.

This backward integration is initialized by taking a quadratic approximation of the terminal cost ℓ_N and setting the derivatives of the value function:

$$V_{\mathbf{x}}(N) = \ell_{N\mathbf{x}},\ V_{\mathbf{xx}}(N) = \ell_{N\mathbf{xx}}$$

where subscript denotes partial derivatives. In order to compute the integration step, we define the argument of the minimum in (3) (a quantity analogous to the Hamiltonian in continuous time) as a function of perturbations around the i-th $(\mathbf{x}, \mathbf{u})$ pair:

$$\begin{aligned} Q(\delta\mathbf{x}, \delta\mathbf{u}) &= \ell(\mathbf{x}+\delta\mathbf{x}, \mathbf{u}+\delta\mathbf{u}, i) - \ell(\mathbf{x}, \mathbf{u}, i) \\ &\quad + V(\mathbf{f}(\mathbf{x}+\delta\mathbf{x}, \mathbf{u}+\delta\mathbf{u}), i+1) - V(\mathbf{f}(\mathbf{x}, \mathbf{u}), i+1) \end{aligned} \quad (9)$$

and expand to second order

$$\approx \frac{1}{2} \begin{bmatrix} 1 \\ \delta\mathbf{x} \\ \delta\mathbf{u} \end{bmatrix}^{\mathsf{T}} \begin{bmatrix} 0 & Q_{\mathbf{x}}^{\mathsf{T}} & Q_{\mathbf{u}}^{\mathsf{T}} \\ Q_{\mathbf{x}} & Q_{\mathbf{xx}} & Q_{\mathbf{xu}} \\ Q_{\mathbf{u}} & Q_{\mathbf{ux}} & Q_{\mathbf{uu}} \end{bmatrix} \begin{bmatrix} 1 \\ \delta\mathbf{x} \\ \delta\mathbf{u} \end{bmatrix}. \quad (10)$$

The expansion coefficients are[1]

$$Q_{\mathbf{x}} = \ell_{\mathbf{x}} \ + \mathbf{f}_{\mathbf{x}}^{\mathsf{T}} V'_{\mathbf{x}} \tag{11a}$$
$$Q_{\mathbf{u}} = \ell_{\mathbf{u}} \ + \mathbf{f}_{\mathbf{u}}^{\mathsf{T}} V'_{\mathbf{x}} \tag{11b}$$
$$Q_{\mathbf{xx}} = \ell_{\mathbf{xx}} + \mathbf{f}_{\mathbf{x}}^{\mathsf{T}} V'_{\mathbf{xx}} \mathbf{f}_{\mathbf{x}} + V'_{\mathbf{x}} \cdot \mathbf{f}_{\mathbf{xx}} \tag{11c}$$
$$Q_{\mathbf{uu}} = \ell_{\mathbf{uu}} + \mathbf{f}_{\mathbf{u}}^{\mathsf{T}} V'_{\mathbf{xx}} \mathbf{f}_{\mathbf{u}} + V'_{\mathbf{x}} \cdot \mathbf{f}_{\mathbf{uu}} \tag{11d}$$
$$Q_{\mathbf{ux}} = \ell_{\mathbf{ux}} + \mathbf{f}_{\mathbf{u}}^{\mathsf{T}} V'_{\mathbf{xx}} \mathbf{f}_{\mathbf{x}} + V'_{\mathbf{x}} \cdot \mathbf{f}_{\mathbf{ux}}. \tag{11e}$$

Note that the last terms in (11c, 11d, 11e) denote contraction with a tensor. Minimizing (10) WRT $\delta\mathbf{u}$ we have

$$\delta\mathbf{u}^*(i) = \underset{\delta\mathbf{u}}{\operatorname{argmin}} Q(\delta\mathbf{x}, \delta\mathbf{u}) = -Q_{\mathbf{uu}}^{-1}(Q_{\mathbf{u}} + Q_{\mathbf{ux}}\delta\mathbf{x}), \tag{12}$$

giving us an open-loop term $\mathbf{k} = -Q_{\mathbf{uu}}^{-1}Q_{\mathbf{u}}$ and a feedback gain term $\mathbf{K} = -Q_{\mathbf{uu}}^{-1}Q_{\mathbf{ux}}$. Plugging the result back in (10), we have a quadratic model of the Value at time i:

$$\Delta V(i) = \quad -\tfrac{1}{2} Q_{\mathbf{u}} Q_{\mathbf{uu}}^{-1} Q_{\mathbf{u}} \tag{13a}$$
$$V_{\mathbf{x}}(i) = Q_{\mathbf{x}} \ - Q_{\mathbf{u}} Q_{\mathbf{uu}}^{-1} Q_{\mathbf{ux}} \tag{13b}$$
$$V_{\mathbf{xx}}(i) = Q_{\mathbf{xx}} - Q_{\mathbf{xu}} Q_{\mathbf{uu}}^{-1} Q_{\mathbf{ux}}. \tag{13c}$$

Recursively computing the local quadratic models of $V(i)$ and the control modifications $\{\mathbf{k}(i), \mathbf{K}(i)\}$, constitutes the backward pass. The main complication stems from the inversion of $Q_{\mathbf{uu}}$ in (12): while the maximum principle guarantees it to be positive definite at the optimal trajectory, it is often not the case throughout the optimization. This can be solved by applying regularization: $\bar{Q}_{\mathbf{uu}} = Q_{\mathbf{uu}} + \lambda\mathbf{I}$.

Once the backward pass is completed, the *forward pass* computes a new trajectory and control sequence:

$$\hat{\mathbf{x}}(1) = \mathbf{x}(1) \tag{14a}$$
$$\hat{\mathbf{u}}(i) = \mathbf{u}(i) + \mathbf{k}(i) + \mathbf{K}(i)\big(\hat{\mathbf{x}}(i) - \mathbf{x}(i)\big) \tag{14b}$$
$$\hat{\mathbf{x}}(i{+}1) = \mathbf{f}(\hat{\mathbf{x}}(i), \hat{\mathbf{u}}(i)). \tag{14c}$$

Note that $\mathbf{K}(i)$ serves as a time-dependent linear feedback term, using the second-order information to ensure a better forward pass

[1]Dropping the index i, primes denoting the next time-step: $V' \equiv V(i{+}1)$.

REFERENCES

[1] P. Abbeel, A. Coates, M. Quigley, and A. Y. Ng. An application of reinforcement learning to aerobatic helicopter flight. In *Advances in Neural Information Processing Systems 19: Proceedings of the 2006 Conference*, page 1, 2007. ISBN 0262195682.

[2] F. A. Almeida. Waypoint navigation using constrained infinite horizon model predictive control. 2008.

[3] J. Baxter and P. L. Bartlett. Infinite-horizon policy-gradient estimation. *Journal of Artificial Intelligence Research*, 15(4):319–350, 2001.

[4] D. P. Bertsekas. Dynamic programming and suboptimal control: A survey from ADP to MPC. *European Journal of Control*, 11(4-5):310–334, 2005. ISSN 0947-3580.

[5] H. G. Bock and K. J. Plitt. A multiple shooting algorithm for direct solution of optimal control problems. *Proceedings of the 9th IFAC world congress*, 1984.

[6] A. E. Bryson and Y. C. Ho. *Applied optimal control*. Wiley New York, 1975. ISBN 0470114819.

[7] H. Chen and F. Allgwer. A Quasi-Infinite horizon nonlinear model predictive control scheme with guaranteed stability. *Automatica*, 34(10):1205–1217, 1998. doi: 10.1016/S0005-1098(98)00073-9. URL http://www.sciencedirect.com/science/article/B6V21-3VCT7RM-5/2/e304a7046ed4bc72e4b4ff0f03ceabde.

[8] M. Diehl, H. Ferreau, and N. Haverbeke. Efficient numerical methods for nonlinear MPC and moving horizon estimation. *Nonlinear Model Predictive Control*, pages 391–417, 2009.

[9] J. C. Dunn and D. P. Bertsekas. Efficient dynamic programming implementations of Newton's method for unconstrained optimal control problems. *Journal of Optimization Theory and Applications*, 63(1):23–38, 1989. ISSN 0022-3239.

[10] A. Gosavi. A reinforcement learning algorithm based on policy iteration for average reward: Empirical results with yield management and convergence analysis. *Machine Learning*, 55(1):5–29, 2004. ISSN 0885-6125.

[11] Bei Hu and A. Linnemann. Toward infinite-horizon optimality in nonlinear model predictive control. *IEEE Transactions on Automatic Control*, 47(4):679–682, 2002. doi: 10.1109/9.995049. URL http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=995049.

[12] David Harris Jacobson and D. Q. Mayne. *Differential Dynamic Programming*. Elsevier, 1970.

[13] Sridhar Mahadevan. Average reward reinforcement learning: Foundations, algorithms, and empirical results. *Machine Learning*, 22:159–196, 1996.

[14] D. Q. Mayne, J. B. Rawlings, C. V. Rao, and P. O. Scokaert. Constrained model predictive control: Stability and optimality. *Automatica*, 36:789–814, 2000. ISSN 0005-1098.

[15] K. Mombaur. Using optimization to create self-stable human-like running. *Robotica*, 27(03):321–330, 2009. ISSN 0263-5747.

[16] M. Morari and J. H. Lee. Model predictive control: past, present and future. *Computers & Chemical Engineering*, 23(4-5):667–682, 1999. ISSN 0098-1354.

[17] Jun Morimoto and Chris A. Atkeson. minimax differential dynamic programming: an application to robust biped walking. In *advances in neural information processing systems (NIPS) 15*, 2003.

[18] G. Pannocchia, S. J. Wright, and J. B. Rawlings. Existence and computation of infinite horizon model predictive control with active steady-state input constraints. *IEEE Transactions on Automatic Control*, 48(6):1002–1006, 2003. doi: 10.1109/TAC.2003.812783. URL http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=1205193.

[19] L. S. Pontryagin, V. G. Boltyanskii, and R. V. Gamkrelidze. The Theory of Optimal Processes. 110(7), 1956.

[20] A. Schwartz. A reinforcement learning method for maximizing undiscounted rewards. In *Proceedings of the*

Tenth International Conference on Machine Learning, volume 298, page 305, 1993.

[21] S. P. Singh. Reinforcement learning algorithms for average-payoff Markovian decision processes. In *Proceedings of the twelfth national conference on Artificial intelligence*, pages 700–705, 1994. ISBN 0262611023.

[22] Y. Tassa and E. Todorov. Stochastic complementarity for local control of discontinuous dynamics. In *Proceedings of Robotics: Science and Systems (RSS)*, 2010.

[23] R. Tedrake, T. W. Zhang, and H. S. Seung. Stochastic policy gradient reinforcement learning on a simple 3D biped. In *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, volume 3, pages 2849–2854, 2004. ISBN 0780384636.

[24] E. Todorov. Eigenfunction approximation methods for linearly-solvable optimal control problems. In *IEEE Symposium on Adaptive Dynamic Programming and Reinforcement Learning (ADPRL)*, pages 161–168. IEEE, 2009.

[25] K. Wampler and Z. Popović. Optimal gait and form for animal locomotion. *ACM Transactions on Graphics (TOG)*, 28(3):1–8, 2009. ISSN 0730-0301. doi: 10.1145/1531326.1531366. URL http://portal.acm.org/citation.cfm?doid=1531326.1531366.

[26] Andrew Witkin and Michael Kass. Spacetime constraints. In *Proceedings of the 15th annual conference on Computer graphics and interactive techniques - SIGGRAPH '88*, pages 159–168, 1988. doi: 10.1145/54852.378507. URL http://portal.acm.org/citation.cfm?doid=54852.378507.

[27] Jia-chi Wu and Zoran Popović. Terrain-adaptive bipedal locomotion control. *ACM Trans. Graph.*, 29:72:1–72:10, July 2010. ISSN 0730-0301.