2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)
October 25-29, 2020, Las Vegas, NV, USA (Virtual)

Nonlinear model predictive control of hopping model using approximate step-to-step models for navigation on complex terrain

Ali Zamani and Pranav A. Bhounsule

***Abstract*— We consider the motion planning problem of a hopper navigating a terrain comprising stepping stones while optimizing an energy metric. The most widely used approach of discrete searches (e.g., A-star) cannot handle boundary conditions (e.g., end path constraints on position, velocity). However, continuous optimizations can easily deal with the boundary value problem but are not widely used in motion planning because they are computationally intensive and possibly non-convex when one considers the terrain. Here we use a continuous optimization approach within a model predictive control framework. First, we generate a library comprising initial states at an instant in the locomotion cycle (e.g., apex), the controls (e.g., foot placement, amplitude of force), and the states at the same instant at the next step. Next, we fit these step-to-step models with low order polynomials (typically 2nd or 3rd order). Finally, the planner uses these low order step-to-step models to preview a fixed distance ahead and plans the optimal steps and controls. Thereafter, we implement the plan for the first step, followed by replanning. This process continues until the hopper reaches the end of the terrain. The main contributions are low-order polynomial models for fast computation and incorporation of the complex terrain as a cost function.**

I. Introduction

The ability to use discrete footholds to move around makes legged robots superior to wheeled robots on terrains with obstacles and ditches. However, planning and control of legged locomotion on such terrain presents a formidable computational challenge because of the complexity of locomotion dynamics (i.e., continuous and discontinuous dynamics) and the complexity of the terrain. Furthermore, if an objective function such as a cost metric is to be optimized, there is an added computational burden of evaluating multiple feasible solutions to determine the best one. In this work, we address the problem of navigating a series of stepping stones, a benchmark problem in dynamic legged locomotion, while optimizing an objective (e.g., energy, time) while taking into account the robot dynamics during the planning phase.

We consider a realistic scenario that a robot might be subjected to and is shown in Fig. 1. Here the robot can see a fixed distance ahead with an onboard vision sensor. The robot then plans the optimum number of steps and control strategy for that fixed distance ahead (planning). Next, the robot implements the control strategy for a few steps (control). Subsequently, the robot previews the next horizontal distance and the process continues until the robot reaches the end of the terrain. This formulation of the problem is well known as receding horizon control or model predictive control

Dept. of Mechanical and Industrial Engineering, University of Illinois at Chicago, 842 W Taylor St, Chicago, IL 60607, USA. Email: alizamani.mecheng@gmail.com, pranav@uic.edu.
This work was supported by NSF grant IIS 1946282.

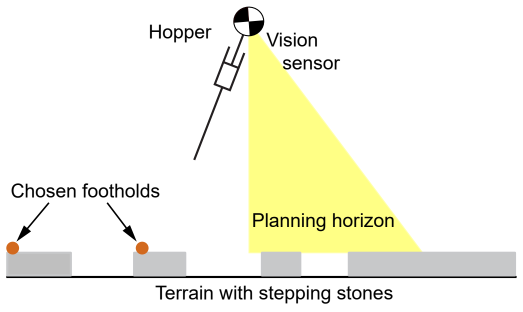

Fig. 1. **Conceptualization of the problem:** The hopper has to negotiate a terrain consisting of stepping stones (gray rectangles). At mid-flight, the hopper plans the optimum strategy and the optimal steps for a planning horizon (yellow patch), then executes the optimum strategy for the first step until the next mid-flight. Then the hopper replans as before continuing the process until it finishes crossing the terrain.

II. Background and Related Work

We limit the review to work done in the area of legged locomotion on complex terrain consisting of obstacles or stepping stones.

A computationally simple approach is to follow a two-tier approach. First, a planner chooses footsteps that enable the robot to go from start to goal and second, a controller that controls the robot to move on those footholds. This method is computationally attractive because the planner only works with the complexity of the terrain and does not consider robot dynamics while the controller is only concerned with the robot dynamics and not the terrain complexity. Chestnutt et. al. [1] used an A-star planner that minimizes heuristics for effort, risk, and/or number and complexity of steps taken to plan footsteps from start to goal. Huang et. al. [2] minimized the energy usage which they parameterized by step length, step width, and step steering in addition to the terrain profile within an A-star planning approach and then solved the footstep planning problem.

The main issue with this two-tiered approach is that the footstep planner does not consider the dynamics of the locomotion and hence there is no guarantee that the plan can actually be implemented. Moreover, since the planner is based on heuristics rather than actual costs (e.g., energy, time), the resulting solution is always sub-optimal.

A more complex but favorable approach is to find all feasible solutions considering the dynamics of the robot. These feasible solutions are enumerated as discrete control actions.

Then using these discrete control actions and with a suitable planner, footholds are planned. Paris et. al. [3] considered the dynamics of the quadruped to map out the reachable states from one step to the next. These reachable states were then searched within an A-star framework to plan jumps on rolling terrain. Campana and Laumond [4] used a similar approach to first map out the feasible motion from step to step followed by footstep planning using probabilistic roadmaps (PRM). The advantage of this approach is that the footstep planner uses feasible solutions only. The disadvantage of using a sampling-based approach like A-star or PRM is that they do not handle boundary conditions that well (e.g., final state specified).

Some other works have focused on finding optimal controllers for a given foot placement locations. Rutschmann et. al. [5] considered the problem of navigating a hopper on a series of pre-assigned footholds using model predictive or receding horizon control. The hopper plans for two steps in succession, but implements only one step and then replanning. Thereafter, the process continues until the hopper reaches the end of the terrain. Nguyen et. al. [6] considered the problem of a biped robot walking on a series of pre-assigned footholds using a control Lyapunov function for biped stability and control barrier function for proper foot placement.

Continuous optimization-based methods overcome boundary value problems. These are optimization formulations where the states and control actions are continuous variables. Continuous optimizations are generally used for trajectory optimization but may be extended to incorporate terrain profile and obstacles. Deits and Tedrake [7], [8] considered the problem of a humanoid robot navigating a series of stepping stones. The continuous optimization formulation was based on kinematic reachability and availability of footholds. The resulting problem is a mixed-integer problem because each foothold needs to be placed only on one of the possible stepping stones. Furthermore, all the nonlinearities were convexified, thus the problem could be solved relatively fast. However, their formulation does not consider the dynamics of the system. More recently Ding et. al. [9] considered the problem of legged robot jumping on an obstacle course. They also formulated a mixed-integer convex program and took into account the reachable space, dynamics, obstacles, and ground reaction forces by suitable approximations to speed up computations.

Our approach is also based on continuous optimization. Like some of the past approaches, we use the step-to-step model to first ascertain the reachable space. This step-to-step model is then approximated by a polynomial model to enable fast computation. The optimization formulation avoids integer variable by specifying a terrain cost that is added onto the cost of locomotion (i.e., the cost of transport, energy used per unit distance moved per unit weight). The resulting optimization problem is solved within a model predictive control framework where the robot scans a fixed distance ahead, then plans footholds, controls, and number of steps, implements the solution for the first step and continues the process until it reaches the end of the terrain. Our novel contributions are: (1) model predictive control framework that incorporates the terrain as a cost avoiding integer constraints, and (2) approximating the step-to-step dynamics with polynomials to enable fast calculations.

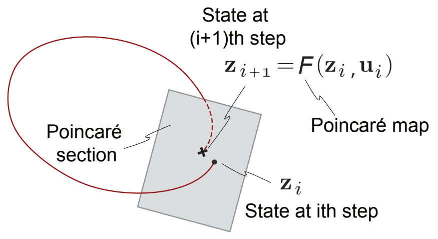

Fig. 2. **Dynamical systems tools use for analysis:** A Poincaré section is an instant in the locomotion cycle (e.g., mid-flight phase for hopper). The state at ith step, $\mathbf{z}_i$ chosen at the Poincaré section. After applying controls $\mathbf{u}_i$ during the step, the system ends at the state $\mathbf{z}_{i+1}$ after one step. There is a function $\mathbf{F}$ that relates the states and controls at step i to the state at step $i+1$.

III. Methods

A. Poincaré section and Poincaré map

We define some preliminaries that are used to analyze and consequently develop controllers for legged systems. Figure 2 (a) (red solid line) shows the trajectory of the robot in state space. We define the Poincaré section as an instant or event in the gait (e.g., the apex event occurs when the vertical velocity is zero during the flight phase for a hopper). We choose an initial condition at the Poincaré section at step i, $\mathbf{z}_i$, and trace its movement on the application of control $\mathbf{u}_i$ for a single step. At the Poincaré section at step $i+1$, the state of the robot is $\mathbf{z}_{i+1}$. There is a function $\mathbf{F}$, known as the step-to-step map or Poincaré map that relates robot state between steps given by

$$\mathbf{z}_{i+1} = \mathbf{F}(\mathbf{z}_i, \mathbf{u}_i). \quad (1)$$

In most cases, there is no analytical solution to the Poincaré map $\mathbf{F}$. Hence once resorts to numerical simulation to obtain $\mathbf{z}_{i+1}$ (output) given the state $\mathbf{z}_i$ and control $\mathbf{u}_i$ (inputs). The numerical integration leads to a computational bottleneck for real-time control.

B. Approximation of the Poincaré map

One of the contributions of this paper is to create and then use an approximation of the Poincaré map ($\overline{\mathbf{F}}$) for control. There are two stages as shown in Fig. 3 to find an approximation of the Poincaré map. First, for data generation, we choose a range of initial states on the Poincaré section ($\mathbf{z}_i$) and a range of control actions in the step ($\mathbf{u}_i$). Next, we use the forward simulation of the system given by $\mathbf{z}_{i+1} = \mathbf{F}(\mathbf{z}_i, \mathbf{u}_i)$, to obtain the numerical value for the system state at the next step $\mathbf{z}_{i+1}$. We show these points as blue dots in Fig. 3 (a). Some of these inputs will lead to a failure, i.e., the state at the next step $\mathbf{z}_{i+1}$ is not defined. We ignore these initial conditions in our curve fitting. Second, for data fitting, we use a suitable defined regression model

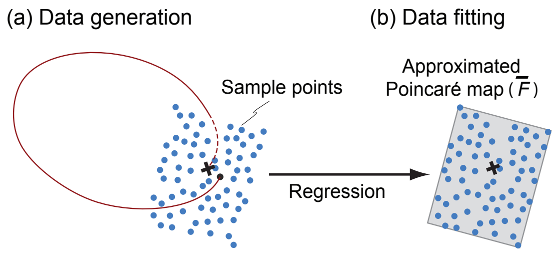

Fig. 3. **Regression to approximate the Poincaré map function $\overline{\mathbf{F}}$:** (a) Data generation: For a range of values of $(\mathbf{z}_i, \mathbf{u}_i)$ at the Poincaré section, we use the forward simulation to generate the robot state at the next step $\mathbf{z}_{i+1}$, shown by blue dots on the left plot. (b) Data fitting: Using an assumed regression model (e.g., 2nd order polynomial), we fit an approximate function to the Poincaré map $\overline{\mathbf{F}}$, i.e., $\mathbf{z}_{i+1} = \overline{\mathbf{F}}(\mathbf{z}_i, \mathbf{u}_i)$, shown by the gray plane.

($\overline{\mathbf{F}}$) to fit the data as shown by the gray plane in Fig. 3 (b). Thus, our approximate model is

$$\mathbf{z}_{i+1} = \overline{\mathbf{F}}(\mathbf{z}_i, \mathbf{u}_i). \tag{2}$$

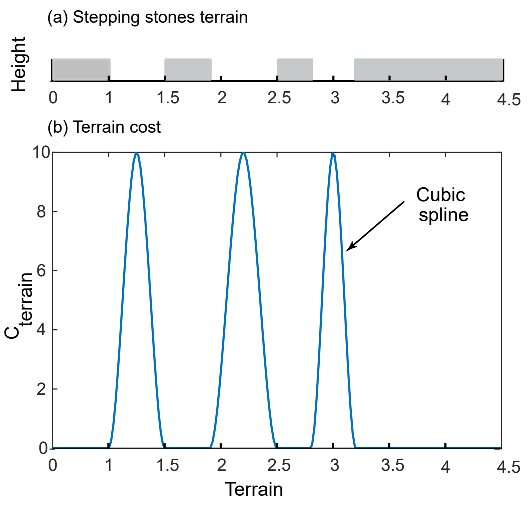

Fig. 4. **Incorporating the terrain profile as a cost:** (a) Stepping stones are shown with gray rectangles and ditches are in the white spaces between the stepping stones. For feasible movement all footholds should be on the stepping stones. (b) We model the terrain with piecewise cubic polynomials. The cost on the stepping stones is zero and increases in the ditches.

C. Stepping stones terrain

Another contribution of the paper is the incorporation of the stepping stones terrain with the optimization. Fig. 4 (a) shows an example terrain consisting of stepping stones. The gray areas are the allowed foothold positions. Previous formulations stipulated the stepping areas as feasible regions and then found foot locations within those feasible regions, thus converting the problem to a mixed-integer problem.

Here we avoid specifying the stepping stones as a constraint, but specify a terrain cost as shown in Fig. 4 (b). We specify a high cost for stepping into the gaps between the stepping stones. We fit a cubic spline to the cost. This formulation is computationally efficient because it does not need to use branch-and-bounds as needed in mixed-integer problems and also allows us to use nonlinear optimization solvers.

D. Non-linear model predictive control problem formulation

The nonlinear model predictive control (MPC) optimization problem uses a nonlinear cost function consisting of the mechanical cost of transport (MCOT) and the terrain cost

$$\begin{aligned}
\min_{N,\mathbf{u}_i,x_{ci}} \quad & \frac{\sum_{i=1}^{i=N} E_i(\mathbf{u_i})}{mg \times x_N} + \sum_{i=1}^{i=N} C_{\text{terrain}}(x_{ci}) \\
\text{subject to:} \quad & \mathbf{z}_{i+1} = \mathbf{F}(\mathbf{z}_i, \mathbf{u}_i) \text{ (or } \overline{\mathbf{F}}) \qquad (3)\\
& \mathbf{u}_{\text{min}} < \mathbf{u}_i < \mathbf{u}_{\text{max}} \\
& x_0 = 0; \\
& x_N = d_{\text{horizon}}; \\
& \dot{x}_0, y_0 \text{ are specified.}
\end{aligned}$$

where $i = 1, 2, ..., N$ is the planning horizon up to N steps noting that N is a decision variable, there are three controls at each step, $\mathbf{u}_i = \begin{bmatrix}\theta & P_b & P_t\end{bmatrix}_i$ thus a total of $3N$ decision variables, and the foot locations x_{ci}, a total of N decision variables. Also, the mechanical energy per step is $E_i = \int(|k(\ell - \ell_0)d\ell| + |P_t d\ell| + |P_b d\ell|)$, $C_{\text{terrain}}(x_{ci})$ is the terrain cost as explained in Sec. III-C, and d_{horizon} is distance the robot can see and thus plan. The absolute value is a non-smooth function of its argument, so we can smooth it out using square-root smoothing [10].

This problem can be solved as a parameter optimization problem. Since the decision variables depend on N, it is not possible to simultaneously optimize N as well as other decision variables. Thus, the optimization has two loops. In the outer loop we optimize N and in the inner loop (for a given N) we optimize the other decision variables. Since planning horizon (d_{horizon}) is short and there are bounds on the maximum and minimum step length, this often leads to solving for only a few number of steps (typically 2 or 3) so it can be done quite fast. We use SNOPT [11] to solve the problem either with the exact model $\mathbf{F}$ or with approximate model $\overline{\mathbf{F}}$. Then we choose the control for the first step and implement on the forward simulation and repeat the calculation until we reach the end of the terrain.

E. Model

We demonstrate our method on a model of hopping shown in Fig. 5 (a) [12]. The model consists of a point mass body m and a massless leg with a maximum leg length ℓ_0. Gravity g points downwards. There is a prismatic actuator that can generate an axial force F along the leg and a hip actuator that can place the swing leg at an angle θ. We non-dimensionalize by dividing the variables with the terms given as follows: distance and leg length by ℓ_0, time by $\sqrt{\ell_0/g}$, velocity by $\sqrt{g\ell_0}$, acceleration by g, and force by mg.

The non-dimensionalized states of the model are given by $\{x, \dot{x}, y, \dot{y}\}$ where x, y are the x- and y- position of the center of mass and $\dot{x}, \dot{y}$ are the respective velocities. A single step of

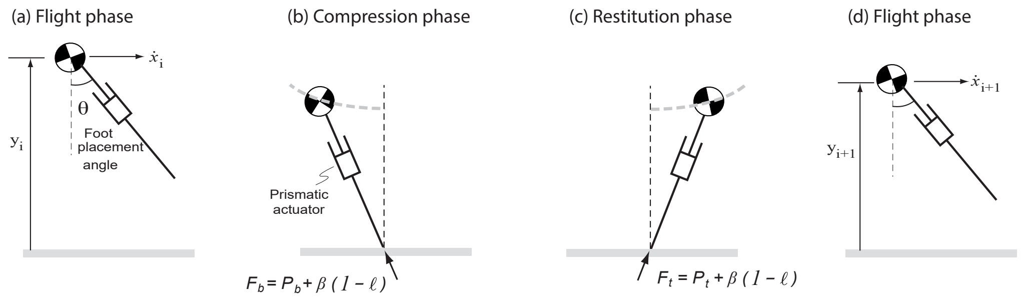

Fig. 5. **A complete step for the hopping model:** The model starts in the flight phase at the apex position (vertical velocity is zero), followed by the stance phase, and finally ending in the flight phase at the apex position of the next step. The hopping model has a prismatic actuator that is used to provide an axial braking force F_b in the compression phase and axial thrust force F_t in the restitution phase, and a hip actuator (not shown) that can place the swinging leg at an angle θ with respect to the vertical as the leg lands on the ground.

the hopper is shown in Fig. 5 (a)-(d). The model starts at the apex (see Fig. 5 (a)) where the state vector is, $\{x_i, \dot{x}_i, y_i, 0\}$. The model then falls under gravity,

$$\ddot{x} = 0, \qquad \ddot{y} = -1 \tag{4}$$

until contact with the ground is detected by the condition $y - \cos\theta = 0$, where θ is the foot placement angle and measured relative to the vertical. Thereafter, the ground contact interaction is given by (see Fig. 5 (b), (c)),

$$\ddot{\ell} = \ell\dot{\theta}^2 - \cos\theta + F \qquad \ell\ddot{\theta} = -2\dot{\ell}\dot{\theta} + \sin\theta \tag{5}$$

$F > 0$ is the non-dimensional linear actuator force along the leg. The first half of the stance phase from touchdown to the maximum compression of the leg length (defined by $\dot{\ell} = 0$) is called the compression phase, the actuator force is $F = P_b + \beta(1-\ell)$ where P_b is constant braking force and $\beta = \frac{k\ell_0}{mg}$. For the second half of the stance phase from mid-stance to take-off, called the restitution phase, the actuator force is $F = P_t + \beta(1-\ell)$ where P_t is constant thrust force. In the above equations $\ell = \frac{\sqrt{(x-x_c)^2+y^2}}{\ell_0}$ is the instantaneous leg length measured relative to the contact point x_c. The take-off phase occurs when the leg is fully extended, that is, $\ell - 1 = 0$. Thereafter, the model has a flight phase and ends up in the next apex state, $\{x^{i+1}, \dot{x}^{i+1}, y^{i+1}, 0\}$ (see Fig. 5 (d)).

IV. Results

A. Computer simulator

We built the forward simulator using MATLAB 2018b. It involves simulating a single step using phases/event stated in Fig. 5 and Eqns. 4 and 5. We integrate these equations using dop853 [13], a higher order Runge-Kutta method with an adaptive step size. The integrator is written as a C file and called by MATLAB through the MEX interface. This makes the simulation reasonably fast. The integrator tolerance is set 10^{-13}. The integrator also has an in-built function *events* to detect a change in phase. Since we treat the simulator as a black-box, we also put checks to detect simulation failure. We consider the robot has failed if it meets any of the following conditions: (1) the horizontal velocity ($\dot{x}$) is negative indicating falling backward; (2) the height of the point mass (y) is below the ground; (3) during take-off from the ground, the vertical velocity is negative ($\dot{y}_{\text{take-off}} < 0$); and (4) at the apex of the flight phase it meets the following condition, $y_{\text{apex-of-flight-phase}} < \ell_0$. The last condition ensures that there is sufficient ground clearance for the swing leg. The only free parameter is $\beta = \frac{kl_0}{mg} = 40$.

B. Polynomial approximation of the Poincaré map & others

a) Data generation: As mentioned in Sec. IV-B we need to approximate the Poincaré map. Our data range for the inputs are: apex horizontal velocity in the range $0.5 \leq \dot{x}_i \leq 2$ in 0.1667 increments; apex height in the range $1.01 \leq y_i \leq 1.5$ in 0.05 increments; foot placement angle in the range $2°$ (0.035 rad) $\leq \theta_i \leq 45°$ (0.7854 rad) in $4.77°$ (0.0833 rad) increments; constant braking force in the range $0 \leq P_{bi} \leq 7$ in 0.7778 increments; and constant thrust force in the range $0 \leq P_{ti} \leq 7$ in 0.7778 increments. This generates $100,000$ input data points $\{\dot{x}_i, y_i, \theta_i, P_{bi}, P_{ti}\}$. For each input data point, we run a forward simulation (see Sec. IV-A) and save the output data, $\{\dot{x}_{i+1}, y_{i+1}\}$. Out of these $100,000$ data points, the robot failed $68,260$ times and successfully took a step $31,740$ times. This means that we only have $31,740$ data points to fit a closed-form expression. We use 75% or $23,805$ successful steps for training and the rest 25% or $7,935$ for testing. Each of the two outputs $\dot{x}_{i+1}$ and y_{i+1} is curve fitted to the inputs, $\mathbf{z}_i = \{\dot{x}_i, y_i\}$ and $\mathbf{u}_i = \{\theta_i \quad P_{bi} \quad P_{ti}\}$.

We also save other auxiliary data such as mechanical energy per step E_i, time from apex to touchdown, time for stance, time from takeoff to apex, and step length.

b) Data fitting: We use a second order polynomial to fit the Poincaré map for each of the two outputs $\dot{x}_{i+1}$ and y_{i+1}. Each polynomial has 21 constants. We use MATLAB function *lsqnonlin* to fit each polynomial. In our testing, we found that 94.2% and 96.1% of the fit for the horizontal velocity and height at the apex, respectively, were within 90% accuracy. It is accepted that 90% accuracy is reasonably good for approximating the model for high fidelity control [14].

In addition, we fit the auxiliary data with a third order polynomial. In our testing, we found that 93.1% fit for mechanical energy, 97% fit for time from apex to touchdown, 93.2% fit for time for stance, 92% for time from takeoff to apex, and 93.7% of the step length was within 90% accuracy.

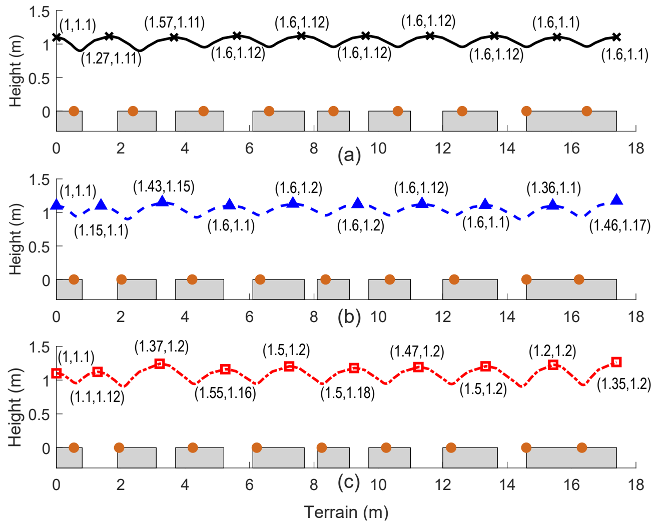

Fig. 6. **Kinematic data for the stepping stones terrain:** (a) Baseline with exact model (b) MPC with exact model, and (c) MPC with approximate model. The stepping stones terrain is shown as gray blocks. The foot placement location is shown as a brown dot on the gray blocks. The trajectory is shown for each of the optimization cases with the apex velocity and apex height at step i marked as $(\dot{x}_i, y_i)$.

C. Optimizations

For all optimizations, the robot starts at apex with initial forward velocity and height of $\dot{x}_0 = 1$ and $y_0 = 1.1$. The stepping stones were randomly assigned as shown in Fig. 6. We did three optimization runs.

a) Baseline optimization: We performed the optimization in Sec. III-D but with $x_N = d_{\text{length}}$ and with the exact Poincaré map $\mathbf{F}$ (i.e., obtained from integration). This is the baseline simulation that we can use to compare our model predictive control results.

b) Model predictive control: We did two model predictive control optimizations. Both of these with $x_N = d_{\text{horizon}} = 4$ m. The only difference is one used the exact Poincaré map $\mathbf{F}$ (MPC with exact model) and the other used the approximate Poincaré map $\overline{\mathbf{F}}$ (MPC with approximate model).

c) Optimization results: Figure 6 shows results for baseline in (a), MPC with exact model in (b), and MPC with approximate model in (c). Each plot shows (1) the number of steps (2) the footstep locations, (3) the vertical height, and (4) the horizontal velocity for the terrain. All three runs produced the same number of steps and approximately the same step locations. However, these solutions differed in the kinematics; the baseline and MPC with exact model had faster speeds and lower jump height compared to MPC with approximate model.

Figure 7 shows the braking force in (a), the thrust force in (b), the foot placement angle in (c), and the Mechanical Cost of Transport in (d) as a function of step number. The braking force is responsible for extracting energy from the system. It can be seen that MPC with approximate model has a higher average force than baseline which is greater than MPC with exact model. The thrust force is responsible for adding

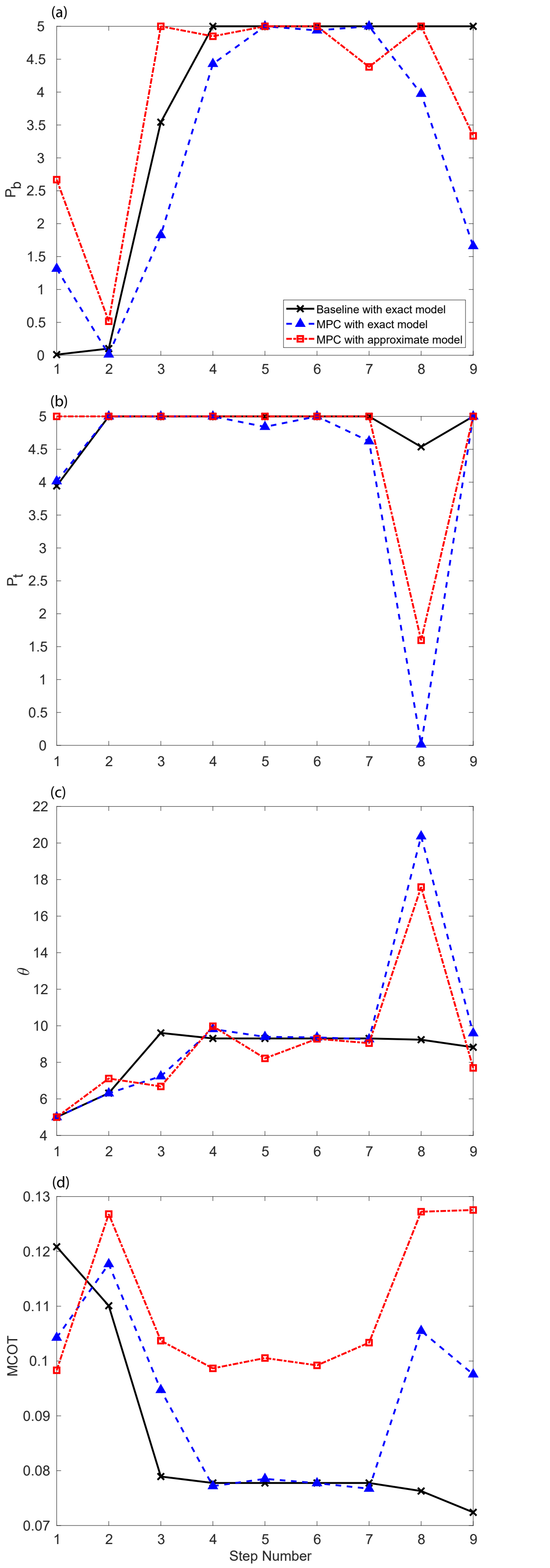

Fig. 7. **Controls and energetics for the stepping stones terrain:** (a) braking force P_{bi} (b) thrust force P_{ti} (c) foot placement angle θ_i (d) Mechanical Cost of Transport MCOT$_i$.

energy to the system. It can be seen that baseline provides on average a greater thrust force than the MPC with approximate model which is greater than MPC with exact model. The foot placement angle is almost the same except at the end where MPC has a larger foot placement angle compared to the baseline. Finally, the MCOT indicates that the baseline is the cheapest after MPC with exact model and finally MPC with approximate model. However, the MPC with approximate model is within 25% of the baseline energetics. These results indicate that MPC formulations give results close to the baseline. Furthermore, the approximate model does reasonably well in terms of control and energetics when compared with baseline and exact models.

Overall, the MCOT for baseline is 0.085, MPC with exact model is 0.092, and MPC with approximate model is 0.11. These results indicate with limited planning such as in MPC the solution is 10% worst than the baseline solution where the entire terrain is known in advance. Also, the MPC with approximate model with 10% modeling error is within 20% of the exact model. We found that optimization time for MPC with approximate model is always faster than MPC with exact model. The speedup is between 2 and 4. This can be notable when real-time planning using MPC is desired.

V. Discussion

In this paper, we have shown that model predictive control (MPC) framework is a viable approach for control of legged locomotion on stepping stones terrain; MPC performs as well as a baseline (ideal) controller that plans the entire terrain before execution in terms of energetics and optimal solution. Furthermore, approximating the step-to-step dynamics with low order polynomials enables faster planning than using the exact model based on integrating the equations of motion.

Our MPC formulation is unique in a few ways. We plan a fixed distance ahead compared to a few steps ahead. The former is the natural choice when we are optimizing the Cost of Transport which is based on distance rather than number of steps. We incorporate the terrain as a cost compared to other formulations where it is incorporated as a constraint. In our cases, the resulting optimal control problem has all real numbers while in the latter there are integers and real values making it a relatively harder optimization [7].

Using a continuous-time based formulation we are able to satisfy the boundary conditions (e.g., position, velocity constraint) which is not possible with sampling-based formulation (e.g., A-star, Rapidly exploring random trees) [15]. However, one caveat is that we need to do multiple optimizations for different step numbers to find the optimal step number. This is usually not a problem with a small preview window as we have in MPC formulations.

MPC is an attractive approach for movement on stepping stones due to its anticipatory nature. However, it is paramount that the computations are done quickly since it is an online method. Since planning the control over continuous time is time consuming we use step-to-step models, computed offline, for online control. These models help to map the reachable space over one step. Another advantage is that these maps are continuous functions of the states and control and thus can be approximated using low order polynomials as done here. Once we have these low order polynomials, real-time planning is feasible. Also, since the plans are computed once per step, they can be evaluated at a lower speed, about half step time, by modest computational resources.

Our approach has some limitations. Our formulation relies on approximating the terrain with a suitable cost function, a heuristic decision. We rely on being able to approximate the Poincaré map and auxiliary variables such as energy, step length, and step time with suitable low order polynomials. When this is not possible one might have to increase the order of the polynomial or resort to other parameterization (e.g., neural network, Gaussian process regression) which may add to the computational burden. Finally, for longer planning horizons the approach may be deemed computationally too intensive preventing the real-time implementation.

References

[1] J. Chestnutt, M. Lau, G. Cheung, J. Kuffner, J. Hodgins, and T. Kanade, "Footstep planning for the honda asimo humanoid," in *Robotics and Automation, 2005. ICRA 2005. Proceedings of the 2005 IEEE International Conference on*. IEEE, 2005, pp. 629–634.

[2] W. Huang, J. Kim, and C. G. Atkeson, "Energy-based optimal step planning for humanoids," in *Robotics and Automation (ICRA), 2013 IEEE International Conference on*. IEEE, 2013, pp. 3124–3129.

[3] V. Paris, T. Strizic, J. Pusey, and K. Byl, "Tools for the design of stable yet nonsteady bounding control," in *2016 American Control Conference (ACC)*. IEEE, 2016, pp. 4822–4828.

[4] M. Campana and J.-P. Laumond, "Ballistic motion planning," in *2016 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2016, pp. 1410–1416.

[5] 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*. IEEE, 2012, pp. 1859–1864.

[6] Q. Nguyen, A. Hereid, J. W. Grizzle, A. D. Ames, and K. Sreenath, "3d dynamic walking on stepping stones with control barrier functions," in *2016 IEEE 55th Conference on Decision and Control (CDC)*. IEEE, 2016, pp. 827–834.

[7] R. Deits and R. Tedrake, "Footstep planning on uneven terrain with mixed-integer convex optimization," in *2014 IEEE-RAS international conference on humanoid robots*. IEEE, 2014, pp. 279–286.

[8] B. Aceituno-Cabezas, J. Cappelletto, J. C. Grieco, and G. Fernández-López, "A generalized mixed-integer convex program for multilegged footstep planning on uneven terrain," *arXiv preprint arXiv:1612.02109*, 2016.

[9] Y. Ding, C. Li, and H.-W. Park, "Single leg dynamic motion planning with mixed-integer convex optimization," in *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2018, pp. 1–6.

[10] P. A. Bhounsule, A. Zamani, J. Krause, S. Farra, and J. Pusey, "Control policies for a large region of attraction for dynamically balancing legged robots: a sampling-based approach," *Robotica*, pp. 1–16, 2020.

[11] P. Gill, W. Murray, and M. Saunders, "SNOPT: An SQP algorithm for large-scale constrained optimization," *SIAM Journal on Optimization*, vol. 12, no. 4, pp. 979–1006, 2002.

[12] A. Zamani and P. Bhounsule, "Control synergies for rapid stabilization and enlarged region of attraction for a model of hopping," *Biomimetics*, vol. 3, no. 3, p. 25, 2018.

[13] E. Hairer, S. NORSETT, and G. Wanner, *Solving Ordinary, Differential Equations I, Nonstiff problems/E. Hairer, SP Norsett, G. Wanner, with 135 Figures, Vol.: 1*. 2Ed. Springer-Verlag, 2000, 2000, no. BOOK.

[14] W. J. Schwind and D. E. Koditschek, "Approximating the stance map of a 2-dof monoped runner," *Journal of Nonlinear Science*, vol. 10, no. 5, pp. 533–568, 2000.

[15] A. Zamani, J. D. Galloway, and P. A. Bhounsule, "Feedback motion planning of legged robots by composing orbital lyapunov functions using rapidly-exploring random trees," in *2019 International Conference on Robotics and Automation (ICRA)*. IEEE, 2019, pp. 1410–1416.