Quadratically constrained quadratic programs using approximations of the step-to-step dynamics: application on a 2D model of Digit

Ernesto Hernandez Hinojosa*, Daniel Torres, and Pranav A. Bhounsule

***Abstract*— Bipedal robots are yet to achieve mainstream application because they lack robustness in real-world settings. One of the major control challenges arises due to the ankle motors' limited control authority, which prevents these robots from being fully controllable at a particular instant (e.g., like an inverted pendulum). We show that to stabilize such robots, they must achieve stability over the time scale of a step, also known as step-to-step (S2S) stability. Past approaches have used the linearization of the S2S dynamics to develop controllers, but these have limited regions of validity. Here, we use a data-driven approach to approximate S2S dynamics, including its region of validity. Our results show that linear and quadratic models can approximate the region of validity and S2S dynamics, respectively. We show that the quadratic S2S approximation generated using a data-driven full-body dynamics simulator outperforms those generated using the analytical linear S2S generated from the popularly used linear inverted pendulum model (LIPM). The S2S approximation enables us to formulate and solve a quadratically constrained quadratic program to develop walking controllers. We demonstrate the efficacy of the approach in simulation using a 2D model of Digit walking on patterned terrain. A video is linked here: `https://youtu.be/MniABg2jGEA`**

I. Introduction

Bipedal robots inspired by animal morphology, such as Digit, seek to mimic the balance, walking control, and flexibility displayed by humans. However, such robots are yet to walk robustly to find their way to practical applications. Bipedal robots are challenging to control due to their high dimensionality, nonlinear dynamics, and under-actuation due to springs and limited ankle torques.

Broadly, two control approaches exist that rely on the fact that walking is, in principle, of low dimensions. One uses a template (e.g., linear inverted pendulum model, LIPM [1]) to develop a controller and then transfer this controller to a full robot model using force and position control. The second develops a trajectory tracking controller using a full-order robot model in such a way that the system behaves like a low-order model (e.g., hybrid zero dynamics [2]). The former does not account for angular momentum or center of mass differences from a point-mass model, while the latter has difficulty stabilizing if the system deviates significantly from the reference trajectory.

Department of Mechanical and Industrial Engineering, University of Illinois at Chicago, 842 W. Taylor St., Chicago, IL 60607 USA. eherna95@uic.edu, dtorre38@uic.edu, pranav@uic.edu. The work is funded by US National Science Foundation through grant 1946282 and 2128568, E.H.H. was supported by the American Heart Association predoctoral fellowship, D.T. was supported by the Illinois Louis Stokes Alliance for Minority Participation Bridge to the Doctorate Fellowship. * Corresponding author

Our approach takes the best of the two. We use a LIPM template to develop the controller and then use tight trajectory tracking on the full order model to reduce it to low dimensions. To circumvent the issue of significant deviation from the nominal trajectory, we use a data-driven approach to identify the state and control deviations that take the system away from the nominal trajectory over the time scale of a step, also known as the S2S dynamics. We fit a quadratic regression model to the S2S dynamics and use the model to develop robust controllers. The model is incorporated as a quadratic constraint in a quadratic program which is used to optimize foot placement to walk over obstacles.

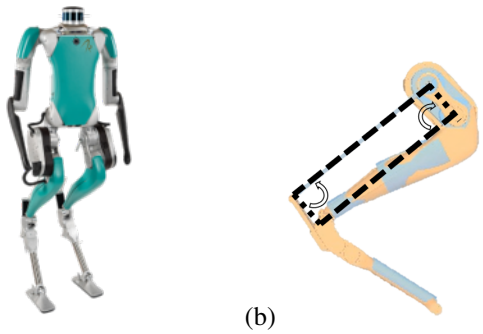
II. Background and Related Work

Control of bipedal robots with small or point feet is challenging because of under-actuation or the lack of control of all degrees of freedom at all times. Such robots are locally unstable, but with a good control design can be stable over the time scale of a step [3]. This notation of step-to-step (S2S) stability is known as orbital stability [4]. There is evidence that humans use S2S stability to balance while walking and running [5].

Some extreme examples of S2S stability are passive dynamic walkers [6]. Here there is no control, and the robot moves down a ramp relying on its natural dynamics [7]. However, such robots are limited to walking down ramps, usually at low speeds, and are knocked out by the slightest disturbances or terrain variation. Adding actuators and actively regulating the torque/force to add just enough energy to overcome friction and energy losses due to foot-strike enables these robots to walk on level ground [8].

To demonstrate S2S or orbital stability, a Poincaré map is used [4]. A Poincaré map relates how the system's state maps from one step to the next. If the eigenvalues of the linearization of the Poincaré map are all less than 1 then the system is said to be S2S or orbitally stable. When control is added, the linearization of the Poincaré map is a function of the state and control. By choosing the control, the eigenvalues can be manipulated to be less than 1 [9].

The Hybrid Zero Dynamics takes a different approach toward S2S stability [10], [11]. Here, virtual constraints are used where the controlled degrees of freedom are tracked as a function of the uncontrolled degree of freedom. This reduces the Poincaré map to a lower dimension (1 for 2D walking and 5 for 3D walking) but does not guarantee S2S stability. To achieve the S2S stability, one could: find controls such that the eigenvalues are less than 1, change the virtual constraints or add event-based stabilization [12]. However,



(a) (b)

Fig. 1: **(a) Robot model:** Digit bipedal robot with 30 degrees of freedom and 20 actuated joints. **(b)** A parallelogram closed chain was used to simplify the 6-bar linkage of the leg.

note that all these methods enable linearized stability, which may be lost for big perturbations.

Since bipedal locomotion is fundamentally a low-dimensional problem, control methods start with an idealized model such as the linear inverted pendulum for walking or the spring-loaded inverted pendulum for running [13]. Then heuristics such as the capture point, which is the location where the robot needs to step to come to a complete stop, may be used to develop control strategies [14]. One of the limitations of such simple models is that they do not capture nonlinearities and changes in angular momentum and may lead to conservative walking when transferred to the full body model or hardware.

Our past work on the spring-loaded inverted pendulum has shown that control based on the nonlinear approximation of the S2S map enables a broader range of perturbations [15]. Then using a 2D humanoid model and approximating the S2S dynamics using polynomials and the region of validity using support vector machines, we were able to demonstrate walking on patterned terrain [16]. Interestingly, it has been shown that approximations to the linear inverted pendulum model (LIPM) can also generate accurate S2S maps for control of the biped Cassie [17]. In this paper, we extend our previous work to the humanoid Digit. Unlike Cassie, Digit has an upper body with a significant mass. We demonstrate that using the LIPM-based S2S map generates poor control compared to using a full nonlinear dynamics-based S2S map. Our contributions are: (1) a method to transfer LIPM models to full body model; (2) use of data-driven approaches to learn the S2S map, including the region of feasibility; and (3) a quadratically constrained quadratic program to plan walking on patterned terrain.

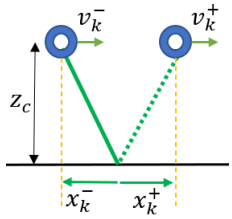
III. Models

A. Robot model

Digit is a 30-degree of freedom bipedal robot with 20 actuated joints shown in Fig. 1(a) (see [18] for more details). Digit's legs are similar to its predecessor Cassie [19], but have an additional roll joint at the toe and an upper body consisting of a torso and two arms. The robot kinematics of the legs consists of two closed-loop chains, modeled as distance constraints between the heel spring and the hip pitch joints. For simplification, the knee and tarsus joint position are inversely coupled during the swing leg phase. This simplification arises because when the heel spring and shin springs are uncompressed, the chain is a parallelogram with two pairs of equal sides as shown in Fig. 1(b).

B. S2S Models

The general form of the S2S models developed in this study is illustrated in Fig. 2 and the general S2S map is shown in Eqn. (1). At the start of the single support phase (SSP), the position and velocity of the center of mass (COM) with respect to the stance foot are x_k^- and v_k^-, respectively, where k is the step number. At the end of the current SSP, the states become x_k^+ and v_k^+. The S2S dynamics model, ζ, maps the states $\mathbf{s_k^-} = \{\mathbf{x_k^-}, \mathbf{v_k^-}\}$ to $\mathbf{s_k^+} = \{\mathbf{x_k^+}, \mathbf{v_k^+}\}$.



$$\mathbf{s_k^+} = \zeta\left(\mathbf{s_k^-}, \tau_\mathbf{p}\right) \quad (1)$$

Fig. 2: The general S2S model.

1) Analytical map from Linear Inverted Pendulum Model (LIPM): In the LIPM the COM height is kept approximately constant using the hip, knee and ankle joints. The equations of the LIPM are linear

$$\ddot{x} = \frac{g}{z_c}x + \frac{1}{mz_c}\tau_p; \qquad \ddot{y} = \frac{g}{z_c}y + \frac{1}{mz_c}\tau_r \quad (2)$$

where g is gravity, z_c is the constant height of the COM, m is the mass, and τ_p and τ_r are input pitch and roll torques, respectively. In the 2D case, assuming no input torques, the equations about the x-axis can be analytically integrated from 0 to time T_s to get

$$\zeta\left(x_k^-, v_k^-\right) = \begin{bmatrix} x_k^{L+} \\ v_k^{L+} \end{bmatrix} = \begin{bmatrix} C_T \cdot x_k^- + T_c S_T \cdot v_k^- \\ S_T/T_c \cdot x_k^- + C_T \cdot v_k^- \end{bmatrix} \quad (3)$$

which fits the form of Eqn. (1) where $S_T = \sinh(T_s/T_c)$, $C_T = \cosh(T_s/T_c)$, and $T_c = \sqrt{z_c/g}$. The state superscripts L and N indicate that the states belong to the LIPM or regression model, respectively.

2) Regression-based map from MuJoCo simulator: A data-derived polynomial regression model was formulated to better capture the S2S dynamics of the robot. The LIPM was used as a starting point to develop a controller for the nonlinear model. The map given by Eqn. (1) was obtained by numerically simulating the system for different initial conditions and controls. This data was then curve fitted using regression analysis. More details are in the Sec. VI. Similar to the LIPM, this model assumes a constant COM height z_c and a constant SSP time T_s of 0.35 seconds. The polynomial S2S map is shown in Eqns. (4) and (5); which also take the form of $\zeta(x_k^-, v_k^-) = \left[\ f(x_k^-, v_k^-), g(x_k^-, v_k^-)\ \right]'$

$$f(x_k^-, v_k^-) = x_k^{N+} = \alpha_0 + \alpha_1 x_k^- + \alpha_2 v_k^- + \alpha_3 x_k^- v_k^- .. \quad (4)$$
$$... + \alpha_4 (x_k^-)^2 + \alpha_5 (v_k^-)^2$$
$$g(x_k^-, v_k^-) = v_k^{N+} = \beta_0 + \beta_1 x_k^- + \beta_2 v_k^- + \beta_3 x_k^- v_k^- ... \quad (5)$$
$$... + \beta_4 (x_k^-)^2 + \beta_5 (v_k^-)^2$$

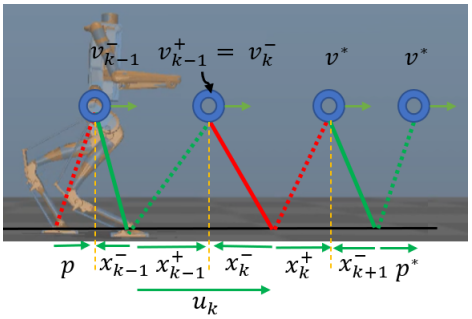


Fig. 3: Model-based stepping. The stepping controller dictates the step size u_k to achieve desired velocity v^*.

IV. Stepping Controllers

The S2S map from the analytical LIPM and regression from nonlinear MuJoCo simulator were used to develop three stepping controllers for stable walking. We assumed that there is no double support phase during walking. The spring constants of the heel and shin were increased to have minimal spring deflection during foot-ground contact. We also assumed that the COM is located at the hip because the arms are fixed and the torso is kept vertically upright. The knee joint of the stance leg is controlled to ensure the position of the hip is at a desired constant height z_c.

A. Model-based Stepping

For this section we will refer to Fig. 3. The current state of the model is (x^-_{k-1}, v^-_{k-1}). The stepping controller is the step length u_k required to achieve the velocity v^* at v^+_k. Therefore the S2S dynamics map can be expressed in terms of the stepping controller

$$\mathbf{s^+_k} = \zeta\left(\mathbf{x^+_{k-1}} - \mathbf{x^-_k}, \mathbf{v^-_k}\right) = \zeta\left(\mathbf{u_k}, \mathbf{v^-_k}\right) \tag{6}$$

The state x^+_{k-1} is uncontrollable because the robot is underactuated at the toe pitch joint. Therefore the only controllable parameter at every step is the state x^-_k. The states (x^+_{k-1}, v^+_{k-1}) are unknown at the current robot state. One way of proceeding with this controller is by predicting the states using the S2S dynamics maps ($[\tilde{x}^+_{k-1}, \tilde{v}^-_k] = \zeta(x^-_{k-1}, v^-_{k-1})$) from Eqns. (3) for the LIPM-based analytical model and Eqns. (4) and (5) for the simulator-based regression model. Note the tilde over the states denotes a predicted state.

The next step is solving for x^-_k to achieve the velocity v^* at v^+_k. We can rearrange the S2S equations into Eqn. (7) to solve for x^-_k using the LIPM.

$$x^-_k = \left(v^+_k - C_T \tilde{v}^-_k\right) T_c / S_T \tag{7}$$

Similarly, we can rearrange the S2S equations into a quadratic equation to solve for x^-_k using the regression model

$$x^-_k = -A + 0.5\beta_4 (A^2 - 4\left(\beta_4\right) B)^{0.5} \tag{8}$$

where $A = \beta_1 + \beta_3 \tilde{v}^-_k$ and $B = \beta_5 (\tilde{v}^-_k)^2 + \beta_2 \tilde{v}^-_k + \beta_0 - v^+_k$. The stepping controllers are u^L_k for the LIPM and u^N_k for the regression model

$$u^L_k = \tilde{x}^+_{k-1} - x^-_k = C_T \cdot x^-_{k-1} + T_c S_T v^-_{k-1} - x^-_k \tag{9}$$

$$u^N_k = \tilde{x}^+_{k-1} - x^-_k = f(x^-_{k-1}, v^-_{k-1}) - x^-_k \tag{10}$$

Cyclic States: A cyclic gait is when the state at an instant maps back to itself after a step. The cyclic state x^{*-} and velocity v^* may be solved by setting $v^+_k = v^-_k = v^*$ and solving for $x^-_k = x^{*-}$ in Eqn. (3) for LIPM-based stepping and Eqn. (5) for simulator-based stepping.

B. Model-based Stepping with Feedback

We assume that the model S2S dynamics are different from the robot dynamics by an additive state error term $\boldsymbol{\omega}$

$$\zeta_R = \zeta\left(x^-_k, v^-_k\right) + \boldsymbol{\omega} \tag{11}$$

where ζ_R is the robot S2S dynamics. The predicted states $(\tilde{x}^+_{k-1}, \tilde{v}^+_{k-1})$ will likely not equal the actual states (x^+_{k-1}, v^+_{k-1}).

1) Analytical, LIPM based: To add robustness to the stepping controller a feedback term can be implemented with gains K properly tuned to drive the error S2S dynamics between the model and the robot model to zero

$$u^R = u^L_k + \mathbf{K}(\mathbf{s(t)^R} - \mathbf{s^L_k}) \tag{12}$$

where u^R is the step size realized on the robot, $\mathbf{s^R}$ is the current state vector of the robot and $\mathbf{s^L_k}$ is the state vector of the LIPM-predicted states x^+_{k-1} and v^+_{k-1}. Stable values for K were found as in [20].

2) Regression, simulator based: Similarly, for regression-based stepping, we can add a feedback term with gains K.

$$u^R = u^N_k + \mathbf{K}(\mathbf{s(t)^R} - \mathbf{s^N_k}) \tag{13}$$

Eqn. (13) is a discrete nonlinear controller which can be linearized about an equilibrium point. Using Eqns. (4) and (5), the error S2S dynamics becomes Eqn. (14).

$$\begin{aligned} &\mathbf{e_{k+1}} = \\ &\left[\begin{array}{c} f\left(\left(x^{+R}_k - x^{+N}_k\right) - \left(u^R_k - u^N_k\right), \left(v^{+R}_k - v^{+N}_k\right)\right) \\ g\left(\left(x^{+R}_k - x^{+N}_k\right) - \left(u^R_k - u^N_k\right), \left(v^{+N}_k - v^{+N}_k\right)\right) \end{array}\right] \\ &= \left[\begin{array}{c} f\left(\left(e^x_k\right) - \left(K\left(\mathbf{e_k}\right)\right), \left(e^v_k\right)\right) \\ g\left(\left(e^x_k\right) - \left(K\left(\mathbf{e_k}\right)\right), \left(e^v_k\right)\right) \end{array}\right] \end{aligned} \tag{14}$$

An equilibrium $\mathbf{e_{k+1}} = \mathbf{e_k}$ exists at $(e^{x*}, e^{v*}) = (0, 0)$. Linearizing the error dynamics near the equilibrium point yields

$$\vec{\mathbf{q}}_{k+1} = \mathbf{J}\vec{\mathbf{q}}_k \tag{15}$$

where $\mathbf{J}$ is the Jacobian of the system, $q_k = (e^x_k - e^{x*})$, $r_k = (e^v_k - e^{v*})$, and $\vec{\mathbf{q}}_k = \left[\begin{array}{c} q_k \\ r_k \end{array}\right]$. The system is now in the form $x_{k+1} = \mathbf{A}x_k$. $\mathbf{K}$ can then be solved for such that all the eigenvalues (λ) of $\mathbf{J}$ evaluated at the equilibrium point are $|\lambda| < 1$ making the system asymptotically stable.

C. Footstrike-corrected Stepping

Although the stepping controllers shown in Eqns. (12) and (13) achieve stable walking patterns when properly tuned, there are two terms that accumulate errors. The first term is in the approximated states $\tilde{x}^+_{k-1}$ and $\tilde{v}^+_{k-1}$ and the second term is in the S2S dynamics error of states x^+_k and v^+_k. We introduce a footstrike-corrected stepping controller,

Eqn. (16), and a footstrike-corrected stepping controller with feedback, Eqn. (17), to improve the regression-based S2S approximation.

$$u_k'^{N} = x(t)_{k-1}^{+R} - \tilde{x}_k^- \tag{16}$$

$$u^R = u_k'^{N} + \mathbf{K}\ \left(\mathbf{s(t)_k^R} - \mathbf{s_k^N}\right) \tag{17}$$

In this approach, the step size $u_k'^{N}$ is updated at every time step using the state x_{k-1}^{+R} of the robot. This updated state is then used to solve for $\tilde{x}_k^-$ such that $v_k^+ = v^*$ is achieved. Eqn. (8) is used to solve for $\tilde{x}_k^-$ given v_k^+ and v_k^-. State $\tilde{x}_k^-$ can be expressed in terms of x_k^{-N} plus an error term c_1 that arises from the differences between x_{k-1}^{+R} and x_{k-1}^{+N}.

$$\tilde{x}_k^- = c_1 + x_k^{-N} \tag{18}$$

The value of c_1 will be small for models where x_{k-1}^{+R} is close to x_{k-1}^{+N}. Solving for $x_k^{-R} - x_k^{-N}$ yields Eqn. (21)

$$\cancel{x(t)_{k-1}^{+R}} - x_k^{-R} = \cancel{x(t)_{k-1}^{+R}} - \tilde{x}_k^- + \mathbf{K}\ \left(\mathbf{s(t)_k^R} - \mathbf{s_k^N}\right) \tag{19}$$

$$\tilde{x}_k^- - x_k^{-R} = \mathbf{K}\ \left(\mathbf{s(t)_k^R} - \mathbf{s_k^N}\right) \tag{20}$$

$$x_k^{-R} - x_k^{-N} = c_1 + \mathbf{K}\ \left(\mathbf{s_k^N} - \mathbf{s(t)_k^R}\right) \tag{21}$$

which is used to solve for the error S2S dynamics.

$$\mathbf{e_{k+1}} = \tag{22}$$
$$\begin{bmatrix} f\left(\left(x_k^{-R} - x_k^{-N}\right), \left(v_k^{-R} - v_k^{-N}\right)\right) \\ g\left(\left(x_k^{-R} - x_k^{-N}\right), \left(v_k^{-R} - v_k^{-N}\right)\right) \end{bmatrix}$$
$$= \begin{bmatrix} f\left(\left(\mathbf{K}\left(\mathbf{e_k}\right) - c_1\right), \left(e_k^v\right)\right) \\ g\left(\left(\mathbf{K}\left(\mathbf{e_k}\right) - c_1\right), \left(e_k^v\right)\right) \end{bmatrix}$$

The nonlinear controller can be linearized about the equilibrium point and gains $\mathbf{K}$ can be solved for such that for small values of c_1 the system is stable.

V. Joint-level Control

The joint-level controllers consist of a gravity compensation feed-forward controller and a PD (proportional-derivative) feedback controller for position and velocity control of the leg actuators. The PD controller drives the actuators to the desired joint trajectory. The joint trajectories of the swing leg are generated by running inverse kinematics on the Cartesian space foot trajectory. The stance leg uses the knee actuator to maintain a desired constant height and the hip pitch and roll actuators to keep the torso upright. The toe actuators of the stance foot are given zero torque and the toe actuators of the swing foot keep the foot parallel to the ground. The arm actuators are kept at a fixed configuration.

A. Gravity Compensation

Gravity compensation at the joints is achieved by modeling the robot with a floating base. The equations of the robot's dynamics, and holonomic force constraints are over-defined because they include dependent degrees of freedom due to the closed-loop chain. F_h are the holonomic forces and F_G are the ground reaction forces.

$$M(q)\ddot{q} + C(q,\dot{q}) + G(q) = B\tau_{\theta,s} + J_G^T F_G + J_h^T F_h \tag{23}$$

$$J_h\ddot{q} + \dot{J}_h\dot{q} = 0 \tag{24}$$

The holonomic constraints include a distance constraint on the closed chain of each leg and a position constraint on the stance foot. For simplicity, the toe rod closed chains are modeled as an open chain with actuation at the toe pitch and roll joints. The open-chain dynamics of the legs are projected onto the constrained dynamics to yield the system in minimal coordinates (see [21] for more details). Once the system is in minimal coordinates, we assume the floating base's six virtual joints provide no force or torque, and the shin and heel spring joint torques are calculated using their respective spring constants K_s and deflection x_s. The leg motor torques and constraint forces are solved for using inverse dynamics on the gravity terms similar to what was done in [20].

B. Joint Trajectory and PD control

The biped is modeled as a compass walker with a point mass at the COM and point feet. Inverse kinematics is done on the foot position for proper foot placement. During the stance phase, the toe motors are given zero torque to allow the toe joints to rotate: allowing an inverted pendulum-like behavior. The knee is used to maintain an approximately constant desired COM height, and the hip pitch joint of the stance leg is used to keep the torso vertically upright. In the 2D case, the position of the hip roll and yaw joints are fixed.

A position and velocity trajectory of the swing foot is generated using a 5th-order polynomial. In the vertical direction, the trajectory is split into two such that the foot moves up to a specified step height and down to the floor in T_s seconds. The horizontal trajectories are model-dependent and are expressed next.

1) Model-based stepping: When using model-based stepping, a trajectory is generated at every footstrike event. The final target velocity is zero, and the targeted final position along the x-axis is dictated by the stepping controller output $u_k^{L/N}$.

$$x\left(t\right)_{k+1}^{foot} = \text{traj}\left(u_k^{L/N}\right) \tag{25}$$

2) Model-based stepping with feedback: Two trajectories are generated in the x direction when using a model-based stepping controller with feedback. The first trajectory is generated at every footstrike event using model-based stepping like Eqn. (25). The second trajectory is updated at every time step; where it begins at the starting position of the foot and finishes at the step length dictated by feedback stepping u^R. Both polynomial trajectory equations are summed with time-dependent weights such that the first trajectory has more weight at the beginning of the step and the second trajectory has more weight at the end of the step.

$$x\left(t\right)_{k+1}^{foot} = \left(1 - \frac{t}{T_s}\right)\text{traj}\left(u_k^{L/N}\right) + \left(\frac{t}{T_s}\right)\text{traj}\left(u^R\right) \tag{26}$$

Newton's method is used to solve the inverse kinematics of the foot trajectory during the swing phase. The shin and heel springs are assumed to be stiff and have zero deflection simplifying the closed chain to a parallel four-bar linkage

as shown in Fig. 1(b). Similarly, body Jacobians are used to compute the velocity. The joint-level controller is

$$\tau_m = \tau_g + K_p(\theta_{des} - \theta) + K_d(\dot{\theta}_{des} - \dot{\theta}) \tag{27}$$

where τ_m is the motor torque, τ_g is the gravity compensation torque, θ_{des} and $\dot{\theta}_{des}$ are the desired joint position and velocity, respectively.

VI. Methods

A. Data Collection using simulator

MuJoCo is an advanced simulator for multi-body dynamics and contacts. MuJoCo was used to run 2D forward walking simulations with different walking parameters and collect S2S data. LIPM-based stepping was applied so the robot could achieve several steps without falling. x_k^-, v_k^-, and v^* were varied between trials and for every step taken x_k^-,v_k^-, x_k^+, and v_{k+1}^-were stored parameters. The stepping controller, Eqn. (12), was used to vary the step size and transition to an orbital state at v^*. The robot was allowed to take up to six steps, but the trial could end sooner if a fall or a slip were detected. Such parameters leading to a fall or slip were labeled as infeasible. The resulting feasible/infeasible dataset was used to define a classification boundary using support vector machine classification. The feasible dataset was used to test the accuracy of the LIPM dynamics and develop an improved model using polynomial regression. An overview of the data collection method is shown in video [22] at 0:05.

B. Feasible Boundary from simulator data

A support vector machine (SVM) classification model was used to approximate the boundary of the feasible data set. The SVM binary classification algorithm from the MATLAB stats toolbox with a polynomial kernel function was used to search for the optimal hyperplane that splits the data into two classes, feasible and infeasible. A training set of 693 combinations of x_k^- and v_k^- was obtained from the LIPM-stepping simulation and was used to train the SVM classifier. The decision boundaries of the SVM classifier were approximated using two lines of best fit of the form shown in Eqn. (28) where $\omega_i(x)$ is the left ($i = 1$) or right ($i = 2$) decision boundary line. Eqn. (29) shows the SVM classifier equations.

$$\omega_i(x) = c_i + d_i \cdot x \tag{28}$$

$$h(x_j) = \begin{cases} +1 \text{ (feasible)} & \text{if } c_1 + d_1 \cdot x_j \geqslant \omega_1 \\ & \text{and } c_2 + d_2 \cdot x_j \leqslant \omega_2 \\ -1 \text{ (not feasible)} & \text{if } c_1 + d_1 \cdot x_j < \omega_1 \\ & \text{or } c_2 + d_2 \cdot x_j > \omega_2 \end{cases} \tag{29}$$

C. Polynomial Regression for S2S map from simulator data

Although the LIPM yields a linear set of the S2S equations, there will be incongruences between the LIPM and full nonlinear dynamics. Regression analysis was performed on the simulation data of the robot in MuJoCo to better capture the robot dynamics. The result was a polynomial regression model shown in Eqns. (4) and (5).

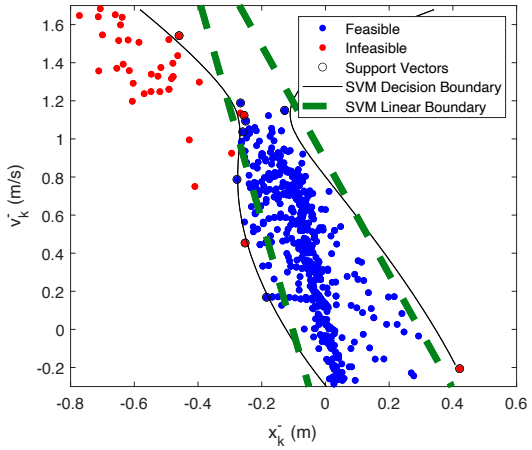


Fig. 4: Feature space and optimal hyperplane of test set.

D. Quadratically Constrained Quadratic Program

An optimization problem was formulated to find the inputs x_k^- and v_k^- that transition the robot to a cyclic gait while minimizing a cost function. The cost function is the squared difference between the outputs and the desired states: Cost $= A\left(\tilde{v}_{k+1}^- - v_{k+1}^-\right)^2 + B\left(\tilde{x}_{k+1}^- - x_{k+1}^-\right)^2$. The weights A and B can be tuned to give more importance to achieving the desired foot placement or COM velocity. The optimal x_k^- is used as the input $\tilde{x}_k^-$ to the model-based stepping controllers. The constraints include the regression equations as well as the linear decision boundary lines. The problem can be expressed as a quadratically constrained quadratic program shown here in matrix form

$$\underset{\mathbf{y}}{\text{minimize}} \quad 0.5\mathbf{y}^T\mathbf{H}\mathbf{y} + \mathbf{f}^T\mathbf{y} \tag{30}$$

$$\text{subject to:} \quad 0.5\mathbf{y}^T\mathbf{Q}_i\mathbf{y} + \mathbf{k}_i^T\mathbf{y} + j_i = 0 \tag{31}$$

$$\mathbf{p}^T\mathbf{y} < b_0, \quad \mathbf{p}^T\mathbf{y} > b_1 \tag{32}$$

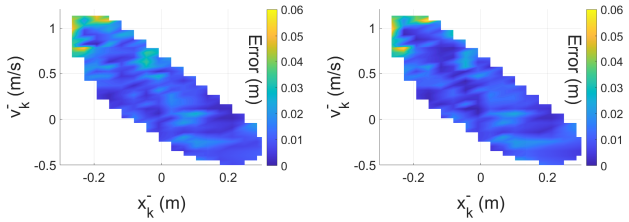
$$\mathbf{LB} \leqslant \mathbf{y} \leqslant \mathbf{UB} \tag{33}$$

where $\mathbf{y} = \begin{bmatrix} x_{k+1}^- & v_{k+1}^- \end{bmatrix}^T$, $\mathbf{f}$, and $\mathbf{H}$ are user chosen constants in the cost function. Eq. (31) accounts for the nonlinear equality constraints composed of the polynomial regression equation Eq. (5). Eq. (32) accounts for the linear inequality constraints composed of the SVM classifier constraints Eq. (29). The lower (LB) and upper bounds (UB) are set to be [-0.5, 0] and [0.6, 1.2], respectively.

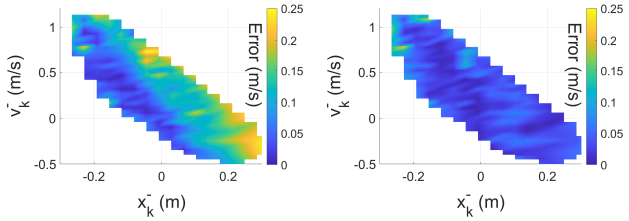
VII. Results

A. Support Vector Machine Classification

Fig. 4 shows the feature space of the training set along with the SVM classification boundary lines where the blue dots are feasible samples and the red dots are infeasible samples. The black line is the SVM optimal hyperplane contour line, and the dashed green line is the approximated linear boundary used as the decision boundary for the model. The overall classification model has an accuracy of 95.8%, and the precision in predicting feasible is 99.7%.



(a) LIPM x_k^+ absolute error (b) Regression x_k^+ absolute error



(c) LIPM v_k^+ absolute error (d) Regression v_k^+ absolute error

Fig. 5: Surface plots of the S2S modelling errors.

B. Quadratic Polynomial Regression

The coefficients of the regression model are $\alpha_{0:5} = [-0.0002, 1.7871, 0.4268, 0, 0.0968, 0]$ and $\beta_{0:5} = [-0.0245, 4.0076, 1.5064, 0.5879, 1.1168, 0.1221]$ all with p-values < 0.01. The adjusted R-squared value for the regression analysis was 0.988 for x_k^+ and 0.987 for v_k^+. The mean absolute error of the regression model was 0.008 m for x_k^+ and 0.0285 m/s for v_k^+ marking an improvement of 22.5% and 60.8%, respectively, over the LIPM. Fig. 5 shows the S2S modeling absolute errors of the test set using the LIPM (left) and the regression model (right). The S2S COM position is captured well using both models. As shown in Fig. 5(a) and Fig. 5(b), both models yield an absolute error of less than 0.02 m for most of the parameter space. The S2S COM velocity is better captured across all of the test set using the regression model as shown in Fig. 5(d).

C. Stepping Control: tracking a reference velocity

We tested the model and controller framework's ability to follow a reference velocity profile $v_{desired}$. The mean absolute error was used as a performance metric; it was calculated by subtracting the reference velocity from the actual at every step and taking the mean. The LIPM stepping controller with no feedback, Eqn. (9), gave a mean absolute error of 25.96% while the regression model stepping controller with no feedback, Eqn. (10), had a mean absolute error of 12.69%; an improvement of 13.27% over the LIPM as shown in Fig. 6a. The LIPM stepping controller with feedback, Eqn. (12), gave a mean absolute tracking error of 22.45% while the regression model stepping controller with feedback, Eqn. (13), gave a mean absolute error of 14.3%; an improvement of 8.15% over the LIPM-based stepping controller with feedback as shown in Fig. 6b. The new stepping controller with footstrike correction, Eqn. (17), yielded a mean absolute error of 9.48%, which is an improvement of 4.82% over the regression stepping controller with feedback (Eqn. (13)) as shown in Fig. 6c.

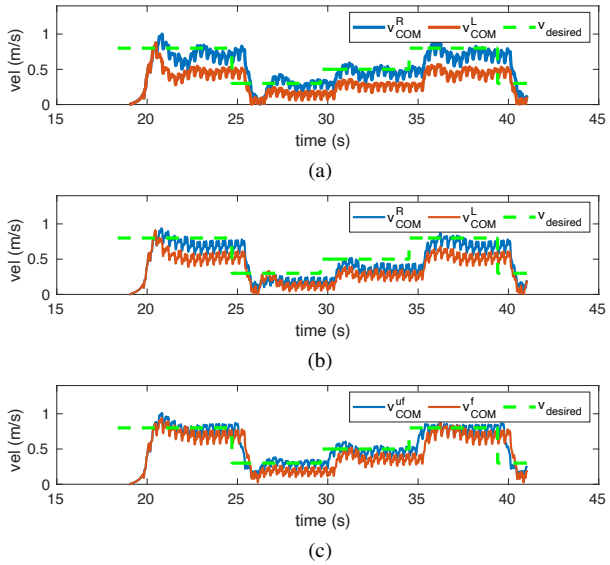


Fig. 6: Velocity tracking using the stepping controller **(a)**, stepping controller with feedback **(b)** and comparison between the LIPM (v^L_{COM}) and regresion model (v^R_{COM}). **(c)**Velocity tracking using regression model with feedback (v^f_{COM}) and footstrike-corrected feedback control (v^{uf}_{COM}).

D. Footstrike-corrected Controller

Using a footstrike-corrected stepping controller reduces the velocity tracking errors, as seen in Fig. 6c. Updating the stepping controller using the robot state x^{+R}_{k-1} as opposed to the model-predicted state $\tilde{x}^{+L/N}_{k-1}$ eliminates the prediction error seen in model-based stepping control. Furthermore, it makes the controller robust to perturbations that the robot may experience during the stance phase. A simulation was conducted where the robot walked forward at steady-state velocity. Two controllers were used: a model-based stepping controller with feedback and a model-based footstrike corrected controller. To test the controller's ability to withstand perturbations while walking, we applied a 175N force for 100ms at the torso along the x-direction (forward direction) in the middle of the stance phase. The force caused the robot dynamics to deviate from the S2S dynamics prediction.

The COM position x_{com} and footstrike state x_k^- are shown in Fig. 7. The footstrike COM velocity state v_k^- is shown in Fig. 8. The perturbation occurs at point (a) causing v_k^- of the next step to increase; this is illustrated by points (b) and (d). However, at point (b), the footstrike-corrected controller has updated the model's state and takes a longer step x_k^-. Meanwhile, at point (d), the model-based controller does not update the model and takes the step size x_k^- it would normally take without a perturbation. The result is that at (c), the longer step of the footstrike-corrected controller has reduced the velocity; conversely, at point (e), a short step causes the COM velocity to spike, causing the robot to become unstable [22](video at 1:26).

E. Optimization: Stepping over obstacles

We tested the model and optimization framework's ability to plan motion on terrain with floor obstacles. The optimiza-

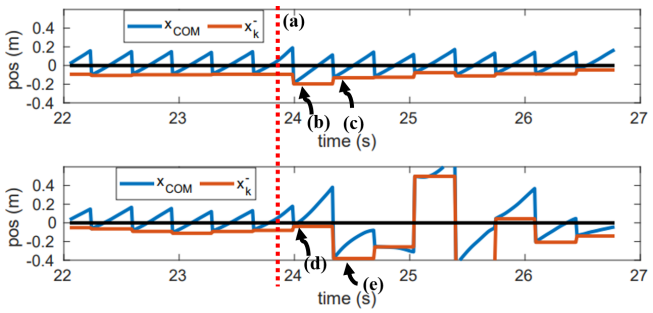


Fig. 7: Position data of COM during perturbation trial using a footstrike-corrected controller (Top) and a model-based controller (Bottom).

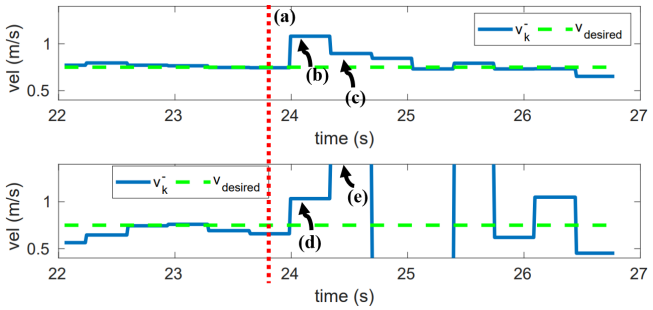


Fig. 8: Velocity data of COM during perturbation trial using a footstrike-corrected controller (Top) and a model-based controller (Bottom).

tion's cost gains vary depending on the robot's environment and the task requirements. Bigger weights are given to velocity cost when foot placement is not critical; such is the case when no obstacles are ahead and a given walking velocity is desired. When foot placement is vital, bigger weights are given to position cost; such is the case when obstacles are ahead and the robot must precisely place its feet to avoid tripping. When the robot is closer than 1.5 m from the obstacle, the larger gains are switched from velocity to foot placement.

An additional linear constraint is added to the optimization problem constraining the footstrike location to be before or after the obstacle. One issue with this formulation is that we need to solve multiple quadratic programs for the footstep planning, one for each feasible footstep location encompassed by obstacles on either side. The choice of the obstacles was such that only two quadratic programs had to be solved, one for stepping before the obstacle and one after the obstacle. After solving both problems, the solution with the lowest cost was chosen as $\tilde{x}_k^-$ in the regression model-based stepping controller with feedback (Eqn. (13)). The control $\tilde{x}_k^-$ was bounded by the feasibility boundary lines and the training set boundary, limiting the robot's ability to take longer or shorter steps when necessary. The gains of the cost function were varied accordingly to give priority to velocity or foot placement as the robot approached the obstacles.

Fig. 9 shows a step-by-step simulation of Digit successfully walking over two obstacles using the regression model stepping controller with feedback (see video at 2:31 in Ref. [22]). Fig. 10 illustrates the states of the robot during the simulation where the time point (a) is the location of the first obstacle and the time point (b) is the location of the

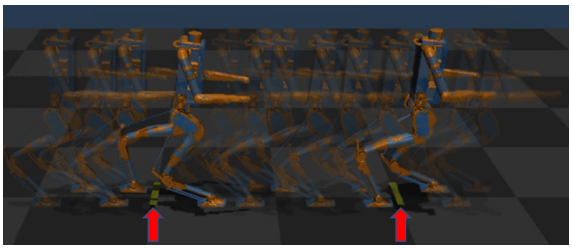


Fig. 9: Simulation of Digit avoiding a trip by stepping over two obstacles (indicated by the red arrow).

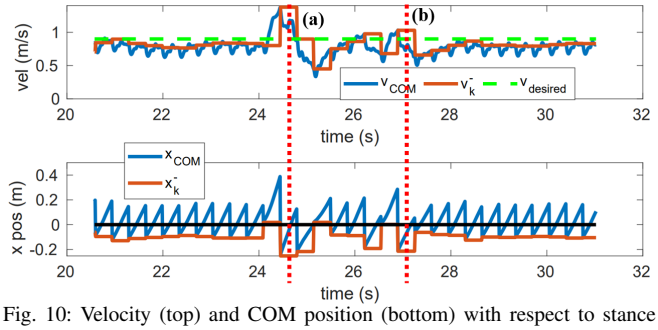


Fig. 10: Velocity (top) and COM position (bottom) with respect to stance foot of Digit while walking over obstacles.

second obstacle. From the figure, we see that Digit takes a very short step before stepping over the obstacle; this causes v_k^- of the following step to increase. Subsequently, a larger step is taken to clear the obstacle. The robot transitions back to the desired steady-state velocity momentarily before encountering the next obstacle. Again, it takes a very small step almost in line with the other foot before taking a longer step to clear the obstacle.

VIII. Discussion

A polynomial regression model that better predicts the S2S dynamics of the robot was created via 2D forward walking simulations using LIPM-based stepping controllers. Although the LIPM is a good one-size-fits-all approach to modeling bipedal walking, it assumes a point mass which may produce modeling errors for heavy bipeds with heavy torsos and swinging arms like Digit. Modeling errors can be fixed at the controller level using an integral feedback term or a state-dependent bias. We proposed an alternative solution: using simulated data from the region near the periodic gait to fit the S2S dynamics of the robot into a polynomial regression model to obtain a wider control region. The model was extrapolated using only two parameters: footstrike position and COM velocity at footstrike. Adding more parameters to the model, such as angular momentum, can further enhance the model. Although the model was tested in the 2D case, the methodology can be extrapolated to the 3D case as was done using the LIPM in [17], [19], [20].

The proposed approach captures some of the non-linearities not captured by the LIPM. Comparing the LIPM stepping controllers and regression model stepping controllers shows that the regression model stepping controller performs better at velocity control. Improving the model, therefore, also enhances the controller. One limitation of the current model is that it relies on a constant stepping frequency and COM height. This limits the ability to use

any step width and walking velocity combination, making motion planning and obstacle avoidance challenging. Future work will focus on addressing these limitations.

A footstrike-corrected feedback controller was introduced to reduce the errors in predicting the footstrike states. This controller is more robust than the model-based stepping controller and can withstand certain perturbations occurring at midstance. One limitation that this controller does not address is that sometimes the steps it takes to recover might not lie inside the feasible set. The robot may eventually become unstable and fall. One possible solution is optimizing the foot placement over a window of several steps such that the robot can recover from a perturbation after several steps lying inside the feasible boundaries. Adding more constraints such as kinematic limits may also enhance the control robustness.

The simple quadratic models and linear constraints allowed us to solve the quadratically constrained quadratic program in less than 100 iterations using a sequential quadratic programming (SQP) algorithm; this is promising for real-time control and for further exploring alternative cost functions such as reduction in motor power or cost of transport. A simple optimization problem was formulated for stepping over obstacles. We limited the planning window to one step. Adding more steps for planning may reduce costs, but the sizeable stepping combination would need suitable heuristics to be solved quickly.

IX. Conclusions and Future Work

This paper presented a data-driven approach to fitting the S2S dynamics of Digit into a polynomial regression model using data generated from a LIPM-based control walking simulation. The regression model was used to develop three types of controllers: model-based, model-based with feedback, and footstrike-corrected controllers. The proposed model is more precise at predicting the S2S dynamics of the robot, which in turn improves the performance of the controllers. We formulated a quadratically constrained quadratic program to solve for the optimal control and used SVM modeling to approximate a feasibility boundary line constraint. We demonstrated that all three stepping controllers developed with the regression model display fewer velocity tracking errors than LIPM-based controllers. In addition, we showed that we could use an optimization problem to plan over terrain with obstacles with high accuracy and few iterations. We presented a more precise data-driven approximation of the S2S dynamics of Digit, which in turn improves the model-based controllers compared to the LIPM. A footstrike-corrected stepping controller was presented, further enhancing the model-based controllers by reducing the points of error in the control formulation. Computationally friendly models and controls will be developed with a similar data-driven approach in future work. The evaluation of this approach on Digit will be performed in hardware.

References

[1] S. Kajita and K. Tani, "Study of dynamic biped locomotion on rugged terrain-theory and basic experiment," in *Proc. of the IEEE International Conference on Robotics and Automation, Sacramento, California, USA*, 1991, pp. 741–746.

[2] J. Grizzle, G. Abba, and F. Plestan, "Asymptotically stable walking for biped robots: Analysis via systems with impulse effects," *IEEE Transactions on Automatic Control*, vol. 46, no. 1, pp. 51–64, 2001.

[3] D. Hobbelen and M. Wisse, "Limit cycle walking," *Humanoid Robots Human-like Machines*, pp. 277–294, 2007.

[4] S. Strogatz, *Nonlinear dynamics and chaos*. Addison-Wesley Reading, 1994.

[5] J. B. Dingwell and H. G. Kang, "Differences between local and orbital dynamic stability during human walking," *Journal of biomechanical engineering*, vol. 129, no. 4, pp. 586–593, 2007.

[6] T. McGeer, "Passive dynamic walking," *The International Journal of Robotics Research*, vol. 9, no. 2, pp. 62–82, 1990.

[7] S. Collins, M. Wisse, and A. Ruina, "A three-dimensional passive-dynamic walking robot with two legs and knees," *The International Journal of Robotics Research*, vol. 20, no. 7, pp. 607–615, 2001.

[8] S. Collins and A. Ruina, "A bipedal walking robot with efficient and human-like gait," in *Proceeding of 2005 International Conference on Robotics and Automation, Barcelona, Spain*, 2005.

[9] P. A. Bhounsule, J. Cortell, A. Grewal, B. Hendriksen, J. D. Karssen, C. Paul, and A. Ruina, "Low-bandwidth reflex-based control for lower power walking: 65 km on a single battery charge," *International Journal of Robotics Research*, 2014.

[10] E. Westervelt, J. Grizzle, and D. Koditschek, "Hybrid zero dynamics of planar biped walkers," *IEEE Transactions on Automatic Control*, vol. 48, pp. 42–56, 2003.

[11] A. Hereid, E. A. Cousineau, C. M. Hubicki, and A. D. Ames, "3d dynamic walking with underactuated humanoid robots: A direct collocation framework for optimizing hybrid zero dynamics," in *2016 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2016, pp. 1447–1454.

[12] C. Chevallereau, J. Grizzle, and C. Shih, "Asymptotically stable walking of a five-link underactuated 3-d bipedal robot," *IEEE Transactions on Robotics*, vol. 25, no. 1, pp. 37–50, 2009.

[13] J. Seipel, M. Kvalheim, S. Revzen, M. A. Sharbafi, and A. Seyfarth, "Conceptual models of legged locomotion," in *Bioinspired Legged Locomotion*. Elsevier, 2017, pp. 55–131.

[14] J. Pratt, J. Carff, S. Drakunov, and A. Goswami, "Capture point: A step toward humanoid push recovery," in *2006 6th IEEE-RAS international conference on humanoid robots*. IEEE, 2006, pp. 200–207.

[15] P. A. Bhounsule, M. Kim, and A. Alaeddini, "Approximation of the step-to-step dynamics enables computationally efficient and fast optimal control of legged robots,," in *ASME 2020 International Design Engineering Technical Conferences and Computers and Information in Engineering Conference*. American Society of Mechanical Engineers, 2020.

[16] E. Hernandez-Hinojosa, A. Satici, and P. A. Bhounsule, "Optimal Control of a 5-Link Biped Using Quadratic Polynomial Model of Two-Point Boundary Value Problem," ser. International Design Engineering Technical Conferences and Computers and Information in Engineering Conference, vol. Volume 8B: 45th Mechanisms and Robotics Conference (MR), 08 2021.

[17] X. Xiong and A. Ames, "3-d underactuated bipedal walking via h-lip based gait synthesis and stepping stabilization," *IEEE Transactions on Robotics*, 2022.

[18] G. A. Castillo, B. Weng, W. Zhang, and A. Hereid, "Robust feedback motion policy design using reinforcement learning on a 3d digit bipedal robot," in *2021 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2021, pp. 5136–5143.

[19] X. Xiong and A. D. Ames, "Orbit characterization, stabilization and composition on 3d underactuated bipedal walking via hybrid passive linear inverted pendulum model," in *2019 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2019, pp. 4644–4651.

[20] X. Xiong and A. Ames, "3d underactuated bipedal walking via h-lip based gait synthesis and stepping stabilization," 2021. [Online]. Available: https://arxiv.org/abs/2101.09588

[21] P. Zhou, A. Zanoni, and P. Masarati, "Projection continuation for minimal coordinate set dynamics of constrained systems," in *ECCOMAS Thematic Conference on Multibody Dynamics*. Budapest University of Technology and Economics, 2021, pp. 184–196.

[22] [Online]. Available: https://youtu.be/MniABg2jGEA