2024 IEEE International Conference on Robotics and Automation (ICRA 2024)
May 13-17, 2024. Yokohama, Japan

Convergent iLQR for Safe Trajectory Planning and Control of Legged Robots

James Zhu, J. Joe Payne, and Aaron M. Johnson

***Abstract*— In order to perform highly dynamic and agile maneuvers, legged robots typically spend time in underactuated domains (e.g. with feet off the ground) where the system has limited command of its acceleration and a constrained amount of time before transitioning to a new domain (e.g. foot touchdown). Meanwhile, these transitions can instantaneously change the system's state, possibly causing perturbations to be mapped arbitrarily far away from the target trajectory. These properties make it difficult for local feedback controllers to effectively recover from disturbances as the system evolves through underactuated domains and hybrid impact events. To address this, we utilize the fundamental solution matrix that characterizes the evolution of perturbations through a hybrid trajectory and its 2-norm, which represents the worst-case growth of perturbations. In this paper, the worst-case perturbation analysis is used to explicitly reason about the tracking performance of a hybrid trajectory and is incorporated in an iLQR framework to optimize a trajectory while taking into account the closed-loop convergence of the trajectory under an LQR tracking controller. The generated convergent trajectories recover more effectively from perturbations, are more robust to large disturbances, and use less feedback control effort than trajectories generated with traditional methods.**

***Index Terms*— Legged Robots, Trajectory Optimization, Robust Control**

I. Introduction

Legged robotics research has increasingly focused on enabling highly dynamic and agile motions such as jumping, leaping, and landing [1–4]. Implementing these capabilities reliably would improve legged robot performance in applications such as extraterrestrial or urban environment navigation where jumping up on ledges or leaping across chasms may be necessary. However, jumping and leaping are dangerous maneuvers, with failure often resulting in catastrophic outcomes for the robot.

What makes these actions challenging is that they induce trajectories that are both hybrid and underactuated, which doubly contribute to the difficulty in controlling legged robots. Broadly speaking, a system is hybrid if it undergoes discrete changes in state and/or dynamics [5,6], and it is underactuated if there exists a direction of acceleration in state space that can not be commanded by any valid input [7, Ch. 1.2]. Even when an underactuated system is controllable, driving the system to a desired target state may require significant time and control effort, neither of which may be readily available. For instance, a robot jumping in the air can not arbitrarily choose how much time it has until its feet touchdown on the ground. This means that the controller needs to spend a lot of effort to correct tracking errors prior to touchdown, or else discontinuous, unbounded saltation effects [8] can cause arbitrarily large divergence if incoming errors are not sufficiently mitigated, e.g. with grazing impacts. Increasing control gains is one possible solution to improve stability, though that strategy comes at a large drawback of worsening robustness in the face of modelling errors and uncertainties [9, Ch. 13]. Instead, this work leverages nonlinearities in continuous and hybrid dynamics that make some trajectories easier to stabilize than others, even under equivalent feedback controllers.

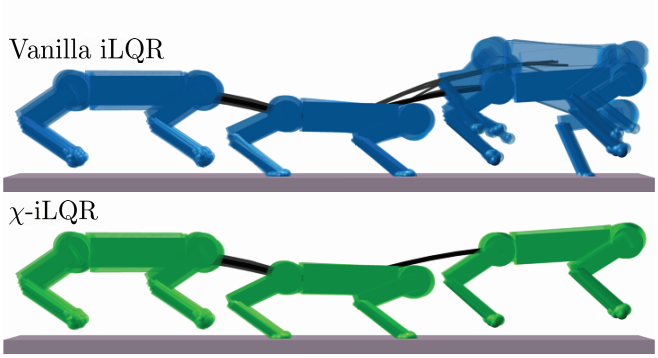

Fig. 1. Similar quadruped gaits tracked with equivalent LQR controllers display enormous differences in final tracking performance. Results for 50 paired trials of trajectories sampled from an initial error covariance of 10^{-2} in all directions are shown. The top blue trajectory was generated with a standard (vanilla) iLQR algorithm, while the bottom green trajectory was generated with our novel χ-iLQR, which improves the average closed-loop convergence by 92%. The horizontal distance between the displayed frames is exaggerated for visual clarity. Only the convergence of the position states is represented, and does not indicate convergence of the velocity states.

This paper presents a novel adaptation of the iLQR trajectory optimization algorithm that improves closed-loop convergence under an equivalent feedback controller (i.e. without changing the LQR controller weights), as demonstrated in Fig. 1. Our simulation results show that this convergent iLQR (χ-iLQR) achieves three simultaneous improvements over standard iLQR: superior tracking performance from initial perturbations, reduced feedback control effort over the trajectory, and improved robustness to large initial errors. Compared to existing methods, χ-iLQR has two additional key strengths. Firstly, it is based on an analysis that is simple to compute compared to methods such as sum-of-squares. Additionally, χ-iLQR captures the local tracking performance of a closed-loop trajectory, which directly predicts experimental results.

This material is based upon work supported by the National Science Foundation under grant #CMMI-1943900.

All authors are with the Department of Mechanical Engineering, Carnegie Mellon University, Pittsburgh, PA, USA, jameszhu@andrew.cmu.edu, amj1@andrew.cmu.edu

II. Related Works

A strategy that has been used to enable dynamic yet precarious behaviors for legged robots is leveraging highly accurate, complex models and full-body trajectory optimization to plan precise motions [3,10]. While these methods incorporate feedback controllers to stabilize the generated trajectories, there has been little focus on how these feedback controllers should be designed to stabilize closed-loop systems under error and uncertainty.

Robust trajectory planning has been successfully implemented for smooth systems like wheeled robots, with some recent results being adapted to hybrid systems like legged robots. These works have focused on optimizing over uncertainties in system dynamics, such as unknown disturbances and modelling errors [11–13]. For example, [11] designs robust closed-loop trajectories for smooth systems by optimizing the volume reduction of an ellipsoidal disturbance set, but was not applied to hybrid systems. Risk-sensitive planning and control is an alternate method that optimizes over the variance of a cost distribution that evolves through the trajectory [14,15]. Other approaches present trajectory optimization algorithms for legged robots over uncertain terrain [16] and compute a forward reachable set to bound closed-loop errors [17]. Many of these methods require the distribution of errors to be prespecified, which is not always clear how to tune. Additional actuation, such as reaction wheels or tails, also relieves the difficulties of underactuated systems [18,19]. However, this comes with obvious tradeoffs of increased cost, size, and weight.

Separately, consider the problem of quantifying the stability or convergence properties of a system. A very popular method is Lyapunov analysis, where the existence of a positive definite differentiable scalar function with negative definite derivatives, called the Lyapunov function, can guarantee asymptotic stability of the system [20]. Lyapunov functions can be difficult to compute, particularly for hybrid systems, and can require methods such as sum-of-squares [21] or machine learning [22] to be tractable. A similar strategy known as control barrier functions, which restricts the system from entering some set of undesirable states, has been successfully implemented on legged robot hardware [23], but has the same drawback as Lyapunov functions.

A different strategy to analyze the stability and convergence of trajectories is contraction analysis [24], which tracks the distance between two close trajectories. If this distance monotonically decreases over the trajectory, then the system is contractive and asymptotic stability can be guaranteed [24]. Contraction analysis has been incorporated into path planning and trajectory optimization algorithms on smooth systems [25,26], but applying contraction analysis to hybrid systems is difficult because many mechanical hybrid systems are not contractive at hybrid events [27]. [28] loosened the contraction criterion and optimized the stability of open-loop periodic orbits using monodromy matrix analysis. Here, we extend that work by generalizing to non-periodic trajectories under feedback control.

III. Analysis and Planning for Hybrid Systems

This section defines a hybrid system and quantifies the performance of a closed-loop hybrid trajectory using linearized variational equations. With this analysis, we can generate a scalar measure of a trajectory's convergence which is then incorporated into a trajectory optimization framework.

A. Hybrid Systems

Hybrid systems are a class of dynamical systems that consist of continuous domains connected by hybrid events [5,6]. Following the notation in [8], we describe a hybrid system as a set of discrete modes $\{\mathrm{I}, \mathrm{J}, \ldots, \mathrm{K}\}$, each with a domain D_{I} and a time-varying vector field F_{I}. $G_{(\mathrm{I},\mathrm{J})}$ is a guard that triggers a transition between mode I and mode J and $R_{(\mathrm{I},\mathrm{J})}$ is the reset map defining that transition.

An execution of a hybrid system [29] begins at an initial state $x_0 \in D_{\mathrm{I}}$. With input $u_{\mathrm{I}}(t, x)$, the system obeys the dynamics F_{I} on D_{I}. If the system reaches guard surface $G_{(\mathrm{I},\mathrm{J})}$, the reset map $R_{(\mathrm{I},\mathrm{J})}$ is applied and the system continues in domain D_{J} under the corresponding dynamics defined by F_{J}. The flow $\phi(t, t_0, x_0, U)$ describes how the hybrid system evolves from some initial time t_0 and state x_0 until some final time t under input sequence U.

B. Linearized Variational Equations

For both continuous domains and hybrid transitions, linearized variational equations can be constructed to characterize the evolution of perturbations δx [30]. In each continuous domain, the linearized variational equation is discretized from timestep i to $i+1$ and is $\delta x_{i+1} \approx (A_{\mathrm{I}} - B_{\mathrm{I}} K_{\mathrm{I}})\delta x_i$ [30] with A_I and B_I being the derivatives of the discretized dynamics in mode I w.r.t. state x_i and control inputs u_i, respectively, and K_{I} are linear feedback gains. The feedback term drops out for open-loop systems. For hybrid events, the analogous variational equation is the saltation matrix $\Xi_{(\mathrm{I},\mathrm{J})}$, which describes the transition between modes I and J. The saltation matrix is the first-order approximation of the change in state perturbations from before the hybrid event at $\delta x(t^-)$ to perturbations after $\delta x(t^+)$ [27], such that $\delta x(t^+) \approx \Xi_{(\mathrm{I},\mathrm{J})} \delta x(t^-)$. This linear approximation assumes that nearby trajectories undergo the same mode transition. Computing the saltation matrix relies on the derivatives of the reset and guard of the transition along with the dynamics in each mode, and is detailed in [8].

C. Convergence Measure

Consider a trajectory that begins at state $x_0 = x(t_0)$ at time t_0 and is executed until time t_f where it arrives at state $x_f = \phi(t_f, t_0, x_0, U)$. The control objective is to bring any nearby initial state $\bar{x}_0 = x_0 + \delta x_0$ towards the nominal trajectory so that at time t_f, $\bar{x}_f = \phi(t_f, t_0, \bar{x}_0, \bar{U}) = x_f + \delta x_f$ is closer to x_f. To characterize the closeness of $\bar{x}_f$ and x_f, we utilize the fundamental solution matrix, Φ. Following [31], the fundamental solution matrix is the linearized approximation $\delta x_f \approx \Phi \delta x_0$ and represents the transformation of error from the initial state to final state.

The fundamental solution matrix can be computed by sequentially composing the linearized variational equations in each continuous domain ($\tilde{A} := A - BK$) and the saltation matrices (Ξ) at each hybrid event [28]. For a hybrid trajectory with N domains, the fundamental solution matrix is:

$$\Phi = \tilde{A}_N \Xi_{(N-1,N)} \dots \Xi_{(2,3)} \tilde{A}_2 \Xi_{(1,2)} \tilde{A}_1 \tag{1}$$

Since the fundamental solution matrix captures the change in errors across a trajectory, the singular values of Φ characterize error change along principle axes of state space. The largest singular value, which is equivalent to the induced 2-norm of Φ, describes the evolution of the most divergent direction of initial error δx_0. We define the convergence measure, χ to be exactly this worst-case value:

$$\chi = ||\Phi||_2 \tag{2}$$

χ is a continuous measure of local convergence, where smaller values of χ indicate stronger reduction of worst-case final errors. A value of $\chi < 1$ indicates errors in all directions will shrink.

D. iLQR for Hybrid Systems

The iterative linear quadratic regulator (iLQR) is a trajectory optimization method that also computes LQR feedback gains over the generated trajectory [32]. iLQR is convenient because compared to other trajectory optimization methods like direct collocation [33], it is less computationally intensive and guarantees a feasible trajectory. We draw from recent work that adapts the iLQR algorithm for use on hybrid dynamical systems [34,35].

In brief, iLQR solves the optimal control problem over N discretized timesteps:

$$\min_{U} \quad \ell_N(x_N) + \sum_{i=0}^{N-1} \ell_i(x_i, u_i) \tag{3}$$

$$\text{where} \quad x_0 = x(0) \tag{4}$$

$$x_{i+1} = \phi(t_{i+1}, t_i, x_i, u_i) \qquad \forall i \tag{5}$$

where $\ell_i(x_i, u_i)$ and $\ell_N(x_N)$ represent the nonlinear stage cost and terminal cost, respectively, $X := \{x_0, x_1, ..., x_N\}$ is a sequence of states with $x_i \in \mathbb{R}^n$ the system state at timestep i and $U := \{u_0, u_1, ..., u_{N-1}\}$ is a sequence of control inputs with $u_i \in \mathbb{R}^m$ the control input at timestep i. We also record the sequence of domains $M := \{D_0, D_1, ..., D_N\}$ with D_i the domain at timestep i such that $x_i \in D_i$. ϕ is the aforementioned flow of the trajectory.

iLQR computes gradient and Hessian information of the cost, which results in a quadratic approximation of the cost function. As such, the state and terminal costs can equivalently be simplified as quadratic functions such that the cost function is simplified to:

$$J = x_N^T Q_N x_N + \sum_{i=0}^{N-1} x_i^T Q_i x_i + u_i^T R_i u_i \tag{6}$$

with $Q_i, Q_N \in \mathbb{R}^{n \times n}$ and $R_i \in \mathbb{R}^{m \times m}$ all positive definite.

iLQR solves the optimal control problem by alternating between forward passes that simulate the system under a given control input sequence, and backward passes that solve for a new locally optimal control sequence. In the backward pass, the value function, which is the optimal cost to go at any timestep, is propagated through the trajectory in reverse, and gives locally optimal feedforward inputs and feedback gains at each timestep. Computing the value function relies on gradient and Hessian computations of the cost function and Jacobians of the dynamics, which equates to computing the linearized variational equations discussed in Sec. III-B. For much greater detail of iLQR for hybrid systems, see [34,35].

IV. Convergent iLQR

Here we present a novel trajectory optimization algorithm called convergent iLQR or χ-iLQR, summarized in Algorithm 1. In this method, the convergence measure is added to the cost function from (6) such that the algorithm minimizes:

$$J_\chi = \quad Q_\chi \chi + x_N^T Q_N x_N + \sum_{i=0}^{N-1} x_i^T Q_i x_i + u_i^T R_i u_i \tag{7}$$

where Q_χ is a scalar weighting parameter. Since χ is solely a function of states and inputs, iLQR uses gradient and Hessian information to make a quadratic approximation compatible with the other cost terms.

Typically in iLQR, the cost function J is evaluated after each forward pass, since it is only dependent on the states and inputs of the most recent trajectory. However, in this case the convergence measure portion of the cost function is dependent on the feedback gains generated by the algorithm. This means that the gradient and Hessian terms of the cost function rely on the feedback gains that are being updated at every timestep in the backward pass. Due to this, the cost function derivatives are highly coupled with the gains and become convoluted to compute.

To resolve this, we propose executing two separate backward passes that each compute a different set of gains. We do this to preserve the convergence properties of iLQR, though other choices might be possible such as borrowing the previous set of gains under the assumption that the gains do not change significantly over iterations. First, the tracking backward pass computes the feedback gains that will be used as the LQR tracking controller gains and to compute the convergence measure. It is equivalent to the backward pass in standard iLQR using the cost function J (6), which solves the Riccati equation for the most recent trajectory. With the gains generated in the tracking backward pass K_t, the convergent cost function J_χ (7) can be computed. The search backward pass takes J_χ from the tracking backward pass and computes the gradients of the convergent cost function with controller gains K_t. The feedforward inputs k_s and the feedback gains K_s from this pass are used to search for an improved trajectory in the forward pass.

Since J_χ is returned by the tracking backward pass, a line search is performed after this function call to guarantee the

Algorithm 1 Convergent iLQR Algorithm

Initialize U, Q_χ, Q_N, Q_i, R_i, $n_{\text{iterations}}$
X, U, M, $J \leftarrow$ ROLLOUT(U)
K_t, J_χ, $P \leftarrow$ TRACKINGBP(X, U, M, J)
$O \leftarrow$ COMPUTEO(X, U, M, K_t)
for $i \leftarrow 1$ to $n_{\text{iterations}}$ **do**
  k_s, $K_s \leftarrow$ SEARCHBP(X, U, M, J_χ, O)
  repeat
    $X, U, M, J, O \leftarrow$ FORWARDPASS(X, U, M, k_s, K_s)
    K_t, J_χ, $P \leftarrow$ TRACKINGBP(X, U, M, J)
  until LINESEARCHISSATISFIED(J_χ)
return X, U, M, K_t

reduction of the cost function J_χ. If the line search condition is not satisfied, the forward pass and tracking backward pass are looped until the line search condition is passed.

Within the search backward pass, iLQR requires computation of the gradient and Hessian of χ. The derivatives of χ can be computed by leveraging the singular value decomposition of $\Phi = USV^T$ where S is a diagonal matrix of singular values and the columns of U and V are the left and right singular vectors, respectively. χ is the largest singular value of Φ and let u_χ and v_χ be its corresponding left and right singular vectors. Following [36], the derivative of χ with respect to the state at timestep i is:

$$\frac{\partial \chi}{\partial x_i} = u_\chi^T \frac{\partial \Phi}{\partial x_i} v_\chi \tag{8}$$

$\frac{\partial \Phi}{\partial x_i}$ in turn can be computed by using the product rule along with leveraging the fact that only $\tilde{A}_i$ and $\Xi_{(i,i+1)}$ are functions of x_i, and all other $\tilde{A}$ and Ξ terms have zero derivatives with respect to x_i. For notational brevity, let:

$$P_i = \tilde{A}_N \cdots \Xi_{(i+1,i+2)} \tilde{A}_{i+1} \tag{9}$$

$$O_i = \Xi_{(i-1,i)} \tilde{A}_{i-1} \cdots \Xi_{(1,2)} \tilde{A}_1 \tag{10}$$

such that $\Phi = P_i \Xi_{(i,i+1)} \tilde{A}_i O_i$ and $\frac{\partial P_i}{\partial x_i} = \frac{\partial O_i}{\partial x_i} = 0$. Thus:

$$\frac{\partial \Phi}{\partial x_i} = P_i \frac{\partial \Xi_{(i,i+1)}}{\partial x_i} \tilde{A}_i O_i + P_i \Xi_{(i,i+1)} \frac{\partial \tilde{A}_i}{\partial x_i} O_i \tag{11}$$

Derivatives with respect to the input u_i follow equivalently. To improve computational efficiency, O_i at each timestep can be computed recursively during the forward pass and each P_i can be computed recursively in the tracking backward pass. Since the rollout does not yet have feedback gain information, the initial O values must be computed separately.

Because the scalar χ is derived from the norm of the matrix Φ, the gradient of χ relies on computing a 3-dimensional tensor of $\tilde{A}_i$ and Ξ derivatives, and the Hessian of χ is computed from a 4-dimensional tensor of matrix second derivatives. While recent work has enabled faster computation of second derivatives of dynamics [37] which can aid in the computation of the 3-D tensor derivatives, computing 4-D tensor derivatives is generally untenable. Instead, numerical methods like finite differences for gradients and BFGS [38] for Hessians can perform at reasonable speed. In order to approach real-time computation, it is likely that the full Hessian of χ is not necessary to find an appropriate search direction and that a partial computation or even leaving out the Hessian completely is sufficient to compute optimal trajectories. Future work will address this gap. Nonetheless, the algorithm in its current form can still be useful for offline planning for trajectories that are expected to have a high degree of risk, such as leaping across ledges or traversing narrow beams. In real-world applications, it can be acceptable for a robot to pause and plan a safe trajectory before executing these dangerous maneuvers.

V. EXAMPLES AND RESULTS

In this section, we demonstrate the convergence improvements of our method on a spring hopper system and a planar quadruped robot model. Simulation results show that the improved convergence measure correlates with an improvement in average tracking performance, robustness to large disturbances, and feedback control effort. Both examples were implemented in MATLAB, with forward simulations using the `ode113` function. Cost function gradients were computed using (11), derivatives of $\tilde{A}_i$ and $\Xi_{(i,i+1)}$ were computed with finite differences, and Hessians were computed with BFGS.

A. Rocket Hopper

1) Rocket Hopper Model: This system is made up of a point mass body with a single massless spring leg. The state of the hopper is characterized by the positions x_B, y_B of the body, the angle θ of the leg and their derivatives $\dot{x}_B$, $\dot{y}_B$, $\dot{\theta}$ such that the full state is a 6×1 vector. The system has two domains: an aerial phase D_1 and a stance phase D_2. Taking a constant ground height at zero gives a touchdown guard function $g_{(1,2)}$ that is the height of the foot and a liftoff guard function $g_{(2,1)}$ that is the ground reaction force applied by the spring leg. Both reset maps $R_{(1,2)}$ and $R_{(2,1)}$ are identity since position and velocity are continuous.

The system has two inputs: a hip actuation and an actuation in the direction of the leg. In the air, this allows the hopper to rotate the leg around the body and exert a propulsion in the direction of the leg, somewhat akin to a rocket, though this force can approximate forces from other legs or actuators. A small rotor inertia in the air ensures the dynamics are well-conditioned when controlling the massless leg. In stance, the hip torque and rocket force exert ground reaction forces on the body. The body mass of the hopper was chosen as 1 kg, spring constant as 250 N/m, and resting leg length as 0.75 m.

2) Rocket Hopper Results: The objective for this system is to begin in the air at rest with a height of 2 m and end in the air at rest with the same height displaced 0.2 m horizontally. The system is given 1.5 s for this trajectory.

We generated four trials of paired trajectories with varied weighting parameters, shown in Table I, and compared the performance of the standard (vanilla) iLQR method (where $Q_\chi = 0$) to χ-iLQR. There is no reference trajectory to track, so Q_i is zero for all trials.

TABLE I

ROCKET HOPPER CONVERGENCE MEASURE AND SIMULATION RESULTS

Trial	LQR Parameters				Convergence Measure			Mean Simulated Error Ratio			Mean Simulated Feedback Effort		
	Q_χ	Q_N	$R_{i_{\text{air}}}$	$R_{i_{\text{stance}}}$	Vanilla	χ-iLQR	%Difference	Vanilla	χ-iLQR	%Difference	Vanilla	χ-iLQR	%Difference
1	50	$500I$	$0.01I$	$0.1I$	1.01	0.71	-29.70%	0.42	0.33	-21.72%	$1.74 \cdot 10^{-5}$	$1.6 \cdot 10^{-5}$	-7.16%
2	50	$800I$	$0.005I$	$0.01I$	0.78	0.51	-34.50%	0.32	0.24	-25.74%	$4.66 \cdot 10^{-5}$	$3.25 \cdot 10^{-5}$	-30.31%
3	50	$250I$	$0.02I$	$0.05I$	1.14	0.94	-17.70%	0.49	0.45	-8.00%	$7.12 \cdot 10^{-6}$	$7.05 \cdot 10^{-6}$	-1.01%
4	75	$500I$	$0.01I$	$0.01I$	1.01	0.68	-33.24%	0.41	0.32	-21.73%	$1.89 \cdot 10^{-5}$	$1.62 \cdot 10^{-5}$	-14.20%

For 3 of these trials, vanilla iLQR generated a trajectory with $\chi > 1$, meaning the worst-case error direction was expansive, see Table I. χ-iLQR decreases every convergence measure to below 1 so that all error directions are reduced over the trajectory. On average, χ-iLQR decreased χ by 28.79% compared to the vanilla method.

To validate these trajectories, each closed-loop trajectory was simulated 100 times with equivalent small random initial perturbations in both positions and velocities with covariance matrix $\text{cov}(X_0) = 10^{-4}I$. A small covariance was chosen so that the linearizations assumed in the convergence measure and LQR control are valid. For each simulation run, the initial error δx_0 and the final error δx_f were recorded, along with the sequence of control inputs $V := \{v_0, v_1, ..., v_{N-1}\}$. Note that these inputs are distinct from the nominal feedforward inputs to the system U because there is additional feedback effort exerted by the actuators.

Two values were recorded during each simulation run to characterize the convergence properties of the trajectories. The first is the error ratio, $E = \frac{||\delta x_f||_2}{||\delta x_0||_2}$ defined as the ratio of the final error 2-norm to the initial error 2-norm. A lower error ratio means better tracking performance, and $E < 1$ indicates a net reduction in error on average. The second value is the feedback effort, $F = \sum_{i=0}^{N-1} (v_i - u_i)^2$ which is the sum of squares of the difference between V and U.

Table I shows that the simulation results support our assertion that an improved convergence measure correlates with an improved mean tracking performance and feedback effort. The mean error ratio and feedback effort over the 100 simulations were both lower for trajectories generated with χ-iLQR. The average improvement over the four trials was 19.30% for mean error ratio and 13.17% for feedback effort.

None of the simulated runs had an error ratio greater than one, which is sensible since the worst-case direction occurs with probability zero. However, even if none of the sampled initial errors aligned exactly with the worst-case direction predicted by the fundamental solution matrix, nearby initial error directions still see improvement in convergence, which explains the improvement in mean simulated error ratio.

B. Planar Quadruped

Here we demonstrate the improvements of χ-iLQR on a more complex robot model akin a standard quadruped robot. The model is simplified as a planar quadruped, meaning that all movement occurs in the sagittal plane and the left-right pairs of legs are constrained to move identically.

1) Planar Quadruped Model: In the sagittal plane, we can model the robot with 7 positional states. x_B, y_B, θ_B are the position and orientation of the body. The front and back sets of legs each have two states for the hip angle α_f, α_b and knee angle β_f, β_b. Thus the full state is dimension 14.

This system has four domains: the aerial domain D_1, front stance domain D_2, back stance domain D_3, and full stance domain D_4. The impact guard function is the height of the foot and the guard function for liftoff is the vertical ground reaction force. The dynamics of the robot body in the aerial phase follow ballistic motion, while the legs are simplified to be massless while including the aforementioned rotor inertia. The impact reset map for each foot consists of a discrete update to the hip and knee velocities, while the body states are unchanged due to the massless legs. The liftoff reset map is identity. The input vector for this system is 4-dimensional to actuate the hip and knee joints.

In this model, parallel torsion springs are added to the knee joints. Parallel joint springs have been utilized to mimic tendons found in animals [39] that increase the energy efficiency of legged locomotion [40,41]. Due to the resonance of the natural spring dynamics, controlling these systems requires special care [42,43]. For example, [44] solved for optimal gait timings to leverage resonant spring frequencies. These spring models of legged robots are good candidates for χ-iLQR because the dynamics of the stance phase depend strongly on the leg configuration at touchdown. Thus, a small error in leg states at touchdown can have a large effect on tracking performance.

The inertial and dimensional properties were chosen to match the Ghost Robotics Spirit 40 quadruped. The added torsional knee spring has a spring constant 75 $\text{N} \cdot \text{m} \cdot \text{rad}^{-1}$ and rest angle 1.2 rad.

2) Planar Quadruped Results: The trajectory optimization task for the planar quadruped is to generate a gait with a forward velocity of 0.25 m/s. The robot begins in the air with a body height of 0.3 m. The hip joints begin at an angle of 0.6 rad and the knee joints begin at 1.2 rad. The terminal target state is translated 0.0875 m in the x-direction from the initial state. The trajectory is given 0.35 s to execute. We choose to set a constant input weight of $R_i = 5 \cdot 10^{-4}I$. The terminal weight is $Q_N = 500I$ and the convergence weight for χ-iLQR is $Q_\chi = 1$.

We set up a similar experiment to the prior example, with the addition of simulating over a range of covariance magnitudes. This is done to evaluate the basin of attraction of

TABLE II

MEAN CONVERGENCE RESULTS FOR THE QUADRUPED MODEL FOR VANILLA AND χ-iLQR WITH COVARIANCE MAGNITUDE 10^{-4}

	Vanilla	χ-iLQR	%Difference
Vanilla Cost	65.58	74.39	+13.43%
Convergence Measure	60.52	41.35	-31.68%
Mean Simulated Error Ratio	6.66	4.78	-28.23%
Mean Simulated Feedback Effort	0.016	0.013	-16.56%

each trajectory over larger initial errors that introduce greater nonlinear effects. The two trajectories were evaluated with 6 sets of 100 paired simulation runs with random initial error covariance magnitudes of 10^{-4}, $5 \cdot 10^{-4}$, 10^{-3}, $5 \cdot 10^{-3}$, 10^{-2}, and $5 \cdot 10^{-2}$ in each direction. The lowest covariance magnitude of 10^{-4} approximates local linear behavior well, while $5 \cdot 10^{-2}$ is the maximum magnitude before some trials begin with the robot's feet below the ground.

Table II shows the vanilla cost (6), convergence measure, mean simulated error ratio, and mean simulated feedback effort of the two trajectories at the covariance magnitude 10^{-4}. As expected, the vanilla cost of the convergent trajectory increases since its optimizing for a different cost function, while the convergence measure and simulation values improve. We argue that in dynamic legged locomotion, a costlier nominal trajectory can often be worth an improvement in the trajectory's robustness.

The mean simulated error ratio for the convergent trajectory at this small covariance magnitude was 28.23% less and the mean simulated feedback effort was 16.56% less. Fig. 2 displays a histogram of the error ratio for each of the trials, with the convergent trajectory having improved performance.

As the magnitude of initial errors grows, the performance of the LQR tracking controller becomes worse due to the increase in nonlinear effects. Fig. 3 shows the simulation results for each trajectory over a range of initial error covariance magnitudes. Each pair of lines indicates the success rate of the respective closed-loop trajectories at maintaining error ratios of less than 50, 10, and 5 respectively. An error ratio of greater than 50 is representative of a catastrophic failure, which the vanilla trajectory encounters at a covariance magnitude of $5 \cdot 10^{-4}$, while the convergent trajectory first experiences a failure at covariance magnitude 10^{-2}. This difference in performance suggests the convergent trajectory is more robust to larger initial errors and nonlinearities.

Even with the χ-iLQR convergence improvements, the controller does not reduce errors in all directions. The simulation results show there was usually some error growth, which is reasonable since the body dynamics are fully unactuated in the aerial phase and the system undergoes multiple hybrid events. A combination of higher feedback gains and a global footstep planner could be able to grant this system full convergence. Even so, this work can be valuable to ensure that the system does not diverge too far from its target trajectory between iterations of a global planner.

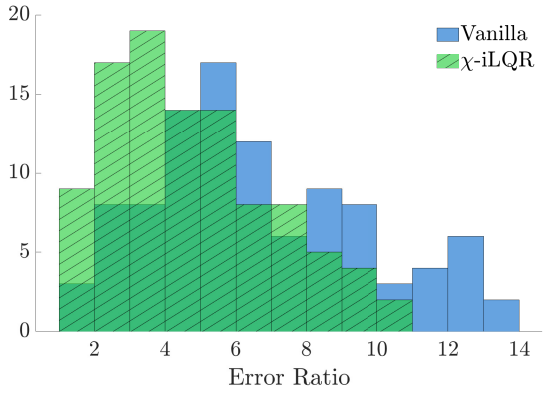

Fig. 2. Histogram of error ratio for 100 paired simulated trials of the quadruped model with small initial perturbations. Error ratio is the 2-norm of final errors divided by the 2-norm of initial errors.

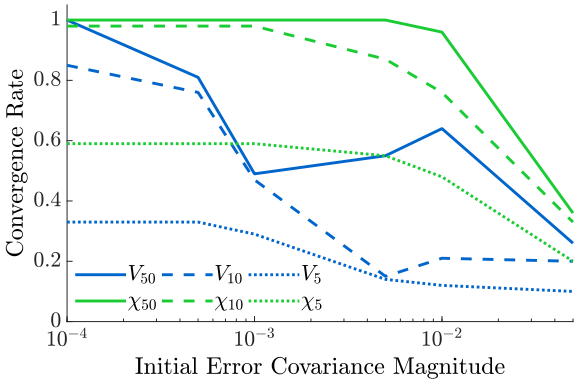

Fig. 3. Plots showing success of LQR controllers at tracking vanilla (V) and convergent (χ) trajectories over various initial perturbation covariances. V_{50} and χ_{50} indicate proportion of trials where error ratio was below 50, V_{10} and χ_{10} below 10, and V_5 and χ_5 below 5.

VI. CONCLUSION

In this work, we present a novel trajectory optimization method, χ-iLQR that optimizes over the worst-case error growth of a hybrid trajectory. This method is based on the fundamental solution matrix, which maps the evolution of perturbations through a trajectory. Incorporating the saltation matrix into the fundamental solution matrix allows for straightforward handling of hybrid events. The simulation results presented on two legged robot models demonstrate that this method produces trajectories with improved tracking performance, decreased feedback actuation effort, and improved robustness to large perturbations. Even for a quadrupedal trajectory that was very difficult to track, χ-iLQR produced a trajectory that was superior at avoiding failures.

Following this work, we aim to apply these principles to robot hardware and demonstrate robust performance for maneuvers such as leaping and flipping. We also aim to apply this work to other hybrid systems like robots that undergo stick-slip transitions. This work and its extensions will further enable robots to navigate complex, uncertain environments and unlock worlds for robots to explore.

REFERENCES

[1] A. M. Johnson and D. E. Koditschek, "Toward a vocabulary of legged leaping," in *IEEE Intl. Conference on Robotics and Automation*, Karlsruhe, Germany, May 2013, pp. 2553–2560.

[2] H. Kolvenbach, E. Hampp, P. Barton *et al.*, "Towards jumping locomotion for quadruped robots on the moon," *IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2019.

[3] C. Nguyen and Q. Nguyen, "Contact-timing and trajectory optimization for 3d jumping on quadruped robots," *IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2022.

[4] Z. Li, X. B. Peng, P. Abbeel *et al.*, "Robust and versatile bipedal jumping control through multi-task reinforcement learning," *arXiv preprint arXiv:2302.09450*, 2023.

[5] A. van der Schaft and J. Schumacher, "Complementarity modeling of hybrid systems," *IEEE Transactions on Automatic Control*, 1998.

[6] J. Lygeros, K. H. Johansson, S. Simic *et al.*, "Dynamical properties of hybrid automata," *IEEE Transactions on Automatic Control*, 2003.

[7] R. Tedrake, *Underactuated Robotics*, 2022. [Online]. Available: http://underactuated.mit.edu

[8] N. J. Kong, J. J. Payne, J. Zhu, and A. M. Johnson, "Saltation matrices: The essential tool for linearizing hybrid dynamical systems," *arXiv preprint arXiv:2306.06862*, 2023.

[9] M. A. Haidekker, *Linear Feedback Controls*, M. A. Haidekker, Ed. Elsevier, 2020.

[10] J. Norby, A. Tajbakhsh, Y. Yang, and A. M. Johnson, "Adaptive complexity model predictive control," *arXiv preprint arXiv:2209.02849*, 2022.

[11] Z. Manchester and S. Kuindersma, "Robust direct trajectory optimization using approximate invariant funnels," *Auton. Robots*, 2019.

[12] J. Morimoto, G. Zeglin, and C. Atkeson, "Minimax differential dynamic programming: application to a biped walking robot," *IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2003.

[13] T. Lew, R. Bonalli, and M. Pavone, "Chance-constrained sequential convex programming for robust trajectory optimization," *European Control Conference*, 2020.

[14] B. Hammoud, M. Khadiv, and L. Righetti, "Impedance optimization for uncertain contact interactions through risk sensitive optimal control," *IEEE Robotics and Automation Letters*, 2021.

[15] M. Ahmadi, X. Xiong, and A. D. Ames, "Risk-sensitive path planning via CVaR barrier functions: Application to bipedal locomotion," *IEEE Control Systems Letters*, 2021.

[16] L. Drnach and Y. Zhao, "Robust trajectory optimization over uncertain terrain with stochastic complementarity," *IEEE Robotics and Automation Letters*, 2021.

[17] S. Kousik, S. Vaskov, F. Bu *et al.*, "Bridging the gap between safety and real-time performance in receding-horizon trajectory design for mobile robots," *The International Journal of Robotics Research*, 2020.

[18] C.-Y. Lee, S. Yang, B. Bokser, and Z. Manchester, "Enhanced balance for legged robots using reaction wheels," *IEEE International Conference on Robotics and Automation*, 2023.

[19] Y. Yang, J. Norby, J. K. Yim, and A. M. Johnson, "Proprioception and tail control enable extreme terrain traversal by quadruped robots," in *IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2023.

[20] H. Khalil, *Nonlinear Systems*. Pearson, 2002.

[21] A. Papachristodoulou and S. Prajna, "On the construction of Lyapunov functions using the sum of squares decomposition," *IEEE Conference on Decision and Control*, 2002.

[22] S. Chen, M. Fazlyab, M. Morari *et al.*, "Learning Lyapunov functions for hybrid systems," *International Conference on Hybrid Systems: Computation and Control*, 2021.

[23] R. Grandia, A. J. Taylor, A. D. Ames, and M. Hutter, "Multi-layered safety for legged robots via control barrier functions and model predictive control," *IEEE International Conference on Robotics and Automation*, 2021.

[24] W. Lohmiller and J. Slotine, "On contraction analysis for non-linear systems," *Automatica*, 1998.

[25] A. M. Johnson, J. E. King, and S. Srinivasa, "Convergent planning," *IEEE Robotics and Automation Letters*, 2016.

[26] N. Kong and A. M. Johnson, "Optimally convergent trajectories for navigation," in *International Symposium on Robotics Research*, October 2019.

[27] S. A. Burden, T. Libby, and S. D. Coogan, "On contraction analysis for hybrid systems," *arXiv preprint arXiv:1811.03956*, 2018.

[28] J. Zhu, N. Kong, G. Council, and A. Johnson, "Hybrid event shaping to stabilize periodic hybrid orbits," *IEEE International Conference on Robotics and Automation*, 2022.

[29] A. M. Johnson, S. A. Burden, and D. E. Koditschek, "A hybrid systems model for simple manipulation and self-manipulation systems," *The International Journal of Robotics Research*, 2016.

[30] M. Hirsch, S. Smale, and R. Devaney, *Differential Equations, Dynamical Systems, and an Introduction to Chaos*. Academic Press, 2012.

[31] R. Leine and H. Nijmeijer, *Dynamics and Bifurcations of Non-Smooth Mechanical Systems*. Springer-Verlag Berlin Heidelberg, 2004.

[32] W. Li and E. Todorov, "Iterative linear quadratic regulator design for nonlinear biological movement systems," *International Conference on Informatics in Control, Automation and Robotics*, 2004.

[33] M. Kelly, "An introduction to trajectory optimization: How to do your own direct collocation," *SIAM Review*, 2017.

[34] N. J. Kong, G. Council, and A. M. Johnson, "iLQR for piecewise-smooth hybrid dynamical systems," *IEEE Conference on Decision and Control*, 2021.

[35] N. Kong, C. Li, G. Council, and A. M. Johnson, "Hybrid iLQR model predictive control for contact implicit stabilization on legged robots," *IEEE Transactions on Robotics*, 2023, to appear. Also available at arXiv:2207.04591 [cs.RO].

[36] J. Townsend, "Differentiating the singular value decomposition," Tech. Rep., 2016. [Online]. Available: https://j-towns.github.io/papers/svd-derivative.pdf

[37] S. Singh, R. P. Russell, and P. M. Wensing, "On second-order derivatives of rigid-body dynamics: Theory & implementation," *arXiv preprint arXiv:2302.06001*, 2023.

[38] C. G. Broyden, "The convergence of a class of double-rank minimization algorithms 1. General considerations," *IMA Journal of Applied Mathematics*, 1970.

[39] K. Endo and H. Herr, "A model of muscle-tendon function in human walking at self-selected speed," *IEEE Transactions on Neural Systems and Rehabilitation Engineering*, 2014.

[40] M. Scheint, M. Sobotka, and M. Buss, "Optimized parallel joint springs in dynamic motion: Comparison of simulation and experiment," *IEEE RAS & EMBS International Conference on Biomedical Robotics and Biomechatronics*, 2010.

[41] K. Mombaur, "Using optimization to create self-stable human-like running," *Robotica*, 2009.

[42] D. Lakatos, K. Ploeger, F. Loeffl *et al.*, "Dynamic locomotion gaits of a compliantly actuated quadruped with slip-like articulated legs embodied in the mechanical design," *IEEE Robotics and Automation Letters*, 2018.

[43] J. Ackerman and J. Seipel, "Energy efficiency of legged robot locomotion with elastically suspended loads," *IEEE Transactions on Robotics*, 2013.

[44] M. J. Pollayil, C. D. Santina, G. Mesesan *et al.*, "Planning natural locomotion for articulated soft quadrupeds," *IEEE International Conference on Robotics and Automation*, 2022.