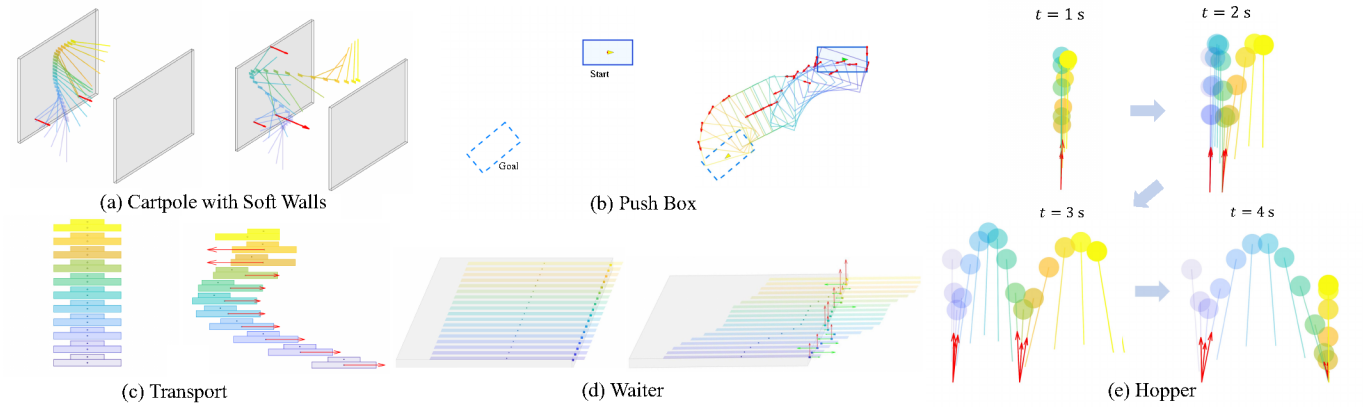
Robotics: Science and Systems 2025
Los Angeles, CA, USA, June 21–25, 2025

On the Surprising Robustness of Sequential Convex Optimization for Contact-Implicit Motion Planning

Yulin Li*,†, Haoyu Han†, Shucheng Kang†, Jun Ma*, and Heng Yang†
*The Hong Kong University of Science and Technology
†Harvard University

https://computationalrobotics.seas.harvard.edu/CRISP



(a) Cartpole with Soft Walls (b) Push Box (c) Transport (d) Waiter (e) Hopper

(f) Real World Push T

Fig. 1: CRISP computes entirely new contact sequences from naive and even all-zero initializations. For (a), (b), (c), and (d), the left side shows the initial trajectories and the right side displays the optimized trajectory from CRISP. For (e) the hopper problem, the initial guess is a free-fall motion released from the origin. The color gradient represents the progression of time (from blue to yellow). For (f), we implement the policy derived from CRISP in a Model Predictive Control (MPC) framework for real-world push tasks. Detailed descriptions of these tasks are provided in §IV.

***Abstract*—Contact-implicit motion planning—embedding contact sequencing as implicit complementarity constraints—holds the promise of leveraging continuous optimization to discover new contact patterns online. Nevertheless, the resulting optimization, being an instance of Mathematical Programming with Complementary Constraints, fails the classical constraint qualifications that are crucial for the convergence of popular numerical solvers. We present robust contact-implicit motion planning with sequential convex programming (CRISP), a solver that departs from the usual primal-dual algorithmic framework but instead focuses only on the primal problem. CRISP solves a convex quadratic program with an adaptive trust region radius at each iteration, and its convergence is evaluated by a merit function using weighted ℓ_1 penalty. We (*i*) prove sufficient conditions on CRISP's convergence to first-order stationary points of the merit function; (*ii*) release a high-performance C++ implementation of CRISP with a generic nonlinear programming interface; and (*iii*) demonstrate CRISP's surprising robustness in solving contact-implicit planning with naive initializations. In fact, CRISP solves several contact-implicit problems with an all-zero initialization.**

I. Introduction

Robots must intelligently establish and disengage *contacts* to successfully perform complex tasks in the physical world, such as manipulation of daily objects and locomotion in rough terrains. However, the hybrid nature of combining continuous dynamics with discrete contact events, the discontinuities in force profiles, and the potential for stick-slip transitions, make it notoriously difficult to "plan through contact".

Contact-implicit motion planning. Among the many efforts for motion planning through contact (see §V for a review), the so-called *contact-implicit* formulation stood out as a particularly popular and promising approach [47, 62]. This formulation distinguishes itself by integrating contact dynamics into the motion planning framework through the introduction of *complementarity constraints*, which allow simultaneous optimization of the state-control trajectories and the contact forces without explicit (and discrete) mode switching (*cf.* (6)). However, the contact-implicit formulation circum-

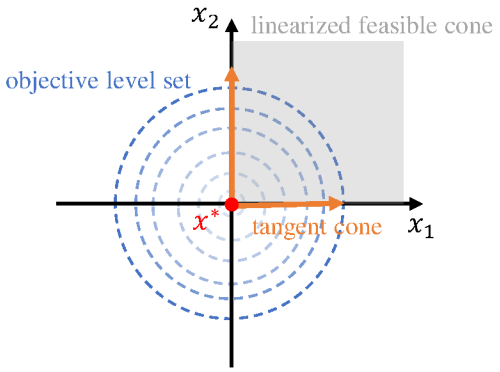


Fig. 2: Geometric intuition of MPCC through Example 1.

vents explicit mode switching at the price of arriving at an optimization problem known as *Mathematical Programming with Complementarity Constraints* (MPCC). Although they "look like" smooth and continuous optimization problems, MPCC problems are known to fail the classical constraint qualifications (CQs) at *all feasible points* (e.g., Linear Independence Constraint Qualification and Mangasarian-Fromovitz Constraint Qualification) [48, 49, 61]. To make this concrete, we provide the geometric intuition through a simple MPCC problem.

Example 1 (Geometric Intuition of MPCC)**.** *Consider the following two-dimensional MPCC problem:*

$$\min_{(x_1,x_2)\in\mathbb{R}^2} \; f(x) := x_1^2 + x_2^2 \tag{1a}$$
$$\text{subject to} \quad g_1(x) := x_1 \geq 0, \tag{1b}$$
$$g_2(x) := x_2 \geq 0, \tag{1c}$$
$$g_3(x) := x_1 \cdot x_2 = 0. \tag{1d}$$

where (1d) *is a complementarity constraint. As depicted in Fig. 2, the objective level sets are circles in* $\mathbb{R}^2$*, and the feasible set consists of the union of the two coordinate axes in the first quadrant*

$$\{x \in \mathbb{R}^2 \mid x_1 \geq 0, x_2 = 0\} \cup \{x \in \mathbb{R}^2 \mid x_1 = 0, x_2 \geq 0\}. \tag{2}$$

The optimal solution of problem (1) *lies at the origin* $(0, 0)$*.*

To understand the local geometry at the origin, we need to characterize two types of "cones" around the origin.

(a) The tangent cone (TC) *at a point is defined as the set of tangent vectors of all curves approaching the point from the feasible region [46]. Thus, the tangent cone at* $(0, 0)$ *is:*

$$TC = \{d \in \mathbb{R}^2 \mid d_1 \geq 0, d_2 = 0\} \cup \{d \in \mathbb{R}^2 \mid d_1 = 0, d_2 \geq 0\}. \tag{3}$$

This consists of two directions: one along the positive x_1*-axis and one along the positive* x_2*-axis, see Fig. 2.*

(b) The linearized feasible cone (LFC) *is formed by all active constraints at that point [46]:*

$$LFC = \left\{ d \in \mathbb{R}^2 \;\middle|\; \begin{array}{l} \nabla g_i^\top d \geq 0, i = 1, 2 \\ \nabla g_j^\top d = 0, j = 3 \end{array} \right\}. \tag{4}$$

Thus, the linearized feasible cone at $(0, 0)$ *is:*

$$LFC = \{d \in \mathbb{R}^2 \mid d_1 \geq 0, d_2 \geq 0\}, \tag{5}$$

i.e., the entire first quadrant. Therefore, we conclude that $TC \subsetneq LFC$*—the TC is a strict subset of the LFC at* $(0, 0)$*.*

We remark that the TC is "geometric", while the LFC is "algebraic". While the definition of LFC involves the algebraic constraints and their linearizations, the definition of TC characterizes the local geometry regardless of how the feasible set is algebraically parameterized.

After noticing the gap between the TC and the LFC, the reader might wonder why this is a problem. The reason lies in that, the equivalence of TC and LFC—typically ensured by CQs—is a prerequisite to establish the Karush–Kuhn–Tucker (KKT) conditions for local optimality [46]. Moreover, almost all primal-dual nonlinear programming (NLP) solvers (e.g., SNOPT [24], IPOPT [57]) rely on the KKT optimality conditions—they search for a pair of primal and dual variables satisfying the KKT conditions. Therefore, the inherent absence of CQs and the gap between TC and LFC of the MPCC problems break the theoretical foundations for primal-dual algorithms. The numerical consequence of this, as we will show in §IV, is that primal-dual solvers can exhibit poor convergence, stuck in infeasible solutions, and/or numerical instabilities when solving contact-implicit MPCC problems.

Relaxation? An extensively studied strategy to mitigate the failure of CQs is to "relax" the complementarity constraint. In [15], using Example 1 as illustration, the authors relaxed the complementarity constraint (1d) as $x_1 \cdot x_2 = \alpha$ for $\alpha > 0$. In [49, 50], a different relaxation scheme where $x_1 \cdot x_2 \leq \alpha$ for $\alpha > 0$ is proposed. In both relaxations, CQs are restored (i.e., no gap between the TC and the LFC) and KKT optimality conditions are reassured. However, since $\alpha = 0$ is the original non-relaxed problem we want to solve, a homotopy scheme is needed to start with α very large and gradually push α to zero. There are two issues with the relaxation approach. First, one needs to solve a sequence of nonconvex problems instead of just one, making this approach computationally expensive. Second, while the problem with α very large is generally well conditioned, as $\alpha \downarrow 0$, numerical issues appear again. For these reasons, to the best of our knowledge, there does not exist a well-accepted implementation of the relaxation approach. Recently in robotics, [33] targeted at linear complementarity constraints—which are indeed KKT optimality conditions of a lower-level convex quadratic program (QP)—and designed a bilevel optimization algorithm where the lower-level QP is solved in a differentiable way to provide gradient for the upper-level trajectory optimization problem. Crucially, to ensure the QP is differentiable with respect to contact, relaxation (or smoothing) is applied again, but this time to the interior point algorithm used for solving the QP. However, (a) it is unclear whether the approach extends to nonlinear complementarity constraints studied in our paper; and (b) the implementation provided by [33] does not follow a generic nonlinear programming interface. Therefore, we did

not benchmark against [33] in §IV.

Can we solve contact-implicit motion planning in a numerically robust way, without relaxation?

Contributions. The answer is rather discouraging if considering a generic nonconvex MPCC problem, as highlighted by the mathematical optimization literature [20, 22, 21]. Nevertheless, contact-implicit planning problems in robotics possess a unique property: the objective function is often *convex* (and quadratic), as the goal typically involves tracking a trajectory or reaching a target. Formally, we consider the problem:

$$\begin{aligned} \min_{v,\lambda} \quad & J(v,\lambda) && \text{(6a)}\\ \text{subject to} \quad & f(v,\lambda)=0, && \text{(6b)}\\ & c_i(v,\lambda)\geq 0,\ \ i\in\mathcal{I} && \text{(6c)}\\ & c_i(v,\lambda)=0,\ \ i\in\mathcal{E} && \text{(6d)}\\ & 0\leq \phi(v,\lambda)\perp \lambda\geq 0, && \text{(6e)} \end{aligned}$$

where v includes the trajectory of both robot state and control inputs, λ denotes the complementarity variables typically associated with contact forces. We assume the objective function $J(v,\lambda)$ is convex—usually a quadratic loss function of tracking errors and control efforts. Constraint (6b) represents the nonlinear system dynamics. Constraints (6c) and (6d) are general inequality and equality constraints. The nonlinear complementarity constraint (6e) enforces the relationship between λ and the nonlinear function $\phi(v,\lambda)$. The expression "$\phi(x,\lambda)\perp\lambda$" should be interpreted as $\phi_i(x,\lambda)\cdot\lambda_i=0$, or its equivalent form $\phi_i(x,\lambda)\cdot\lambda_i\leq 0$ as suggested by [22], for every entry of $\phi(\cdot)$ and λ. It should be emphasized that problem (6) is nonlinear and nonconvex, as we impose no restrictions on the convexity of the constraints. This formulation is classified as a nonlinear complementarity problem (NCP) due to constraints (6e). A key distinction between NCP and linear complementarity problem (LCP) lies in the nonlinearity present in both the complementarity constraints and other constraints. While one can use $s=\phi(v,\lambda)$ to recast complementarity constraints as $\lambda\perp s$, this transformation merely shifts the nonlinearity to the additional constraint $s=\phi(v,\lambda)$. In the linear complementarity problem (LCP) literature, both ϕ and other constraints are typically required to be linear (as in [5] and LCQpow [28]). In contrast, CRISP is capable of handling generic nonlinear dynamics, as demonstrated in our examples. We will give concrete examples of (6) in §IV. For now, let us work with the generic template (6).

Our major contribution is to depart from the usual primal-dual algorithmic framework—due to the failure of CQs in MPCC—and propose a *primal-only* algorithm named CRISP (**R**obust **C**ontact-**I**mplicit motion planning with **S**equential convex **P**rogramming). CRISP features a merit function with weighted ℓ_1 penalty to measure primal feasibility and objective reduction. It solves a series of trust-region convex quadratic programs (QPs) formulated using local second-order information of the objective and *first-order* information of the constraints. In stark contrast with the well-known sequential quadratic programming (SQP) framework, CRISP (a) does not require second-order information of the constraints, and (b) does not maintain and update dual variables. Since the objective function is convex, the inner-loop QP subproblem inherits convexity by construction. These make CRISP both simple to implement and computationally efficient.

One then wonders whether a simple algorithm like CRISP can offer any convergence guarantees. We show that the convexity of the objective function allows us to prove sufficient conditions under which CRISP can guarantee convergence to the stationary points of the merit function. Importantly, the sufficient conditions are numerically verifiable, and hence the convergence can be "certified" from the numerical iterations of CRISP. Moreover, we show that the local minima of the merit function are indeed the local minimizers of the original problem if feasible for the original problem to close the loop.

Finally, we open-source a high-performance C++ implementation of CRISP. Our C++ implementation follows a generic nonlinear programming interface where the user defines objective function and constraints. We leverage automatic differentiation to obtain gradient and Hessian information. We then apply CRISP to solve six contact-implicit motion planning problems and benchmark its performance against both generic and robotics-specific solvers. We believe the way we model some of the contact-implicit planning problems is also new. Our numerical results demonstrate that CRISP consistently generates non-trivial and entirely new contact sequences from naive and even all-zero initializations, while existing solvers can struggle to find even feasible solutions.

To summarize, our contributions are:

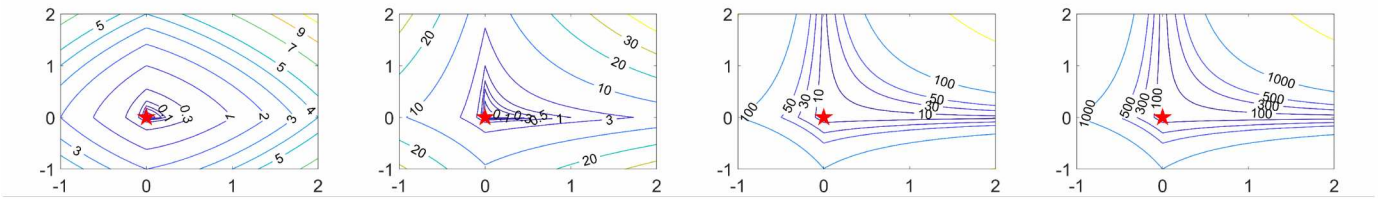
- **Theory and Algorithm**. We propose a primal-only algorithm for contact-implicit motion planning called CRISP and provide theoretical guarantees.
- **Implementation**. We release an open-source high-performance C++ implementation of CRISP.
- **Benchmark**. We model six contact-rich planning problems using the contact-implicit formulation and benchmark CRISP against existing solvers.

Paper organization. We present the theory and algorithm for CRISP in §II, its implementation details in §III, and benchmark results in §IV. We postpone the related work review to §V and conclude in §VI. Appendix and project website provide proofs and the detailed contact-implicit formulation for the five planning problems, as well as the animations of numerically computed motion trajectories.

II. CRISP: Theory and Algorithm

Clearly, the contact-implicit planning problem (6) can be transformed into a generic NLP problem, which also aligns with our implemented C++ solver. Let $x=[v,\lambda]$, we denote the NLP formulation of (6) as:

$$\begin{aligned} \min_{x} \quad & J(x) && \text{(7a)}\\ \text{subject to} \quad & c_i(x)=0, \quad i\in\mathcal{E} && \text{(7b)}\\ & c_i(x)\geq 0. \quad i\in\mathcal{I} && \text{(7c)} \end{aligned}$$



(a) $\mu = 1$ (b) $\mu = 10$ (c) $\mu = 100$ (d) $\mu = 1000$

Fig. 3: Depiction of the ℓ_1 penalty merit function for Example 1 with different penalty parameters μ.

Here, we absorb all other constraints into the general template c for simplicity. **Primal-only merit function.** From a primal-only perspective, the core optimization challenge is to balance descent of the objective function with satisfaction of the constraints [46]. In CRISP, we adopt the following merit function with weighted ℓ_1 penalty to evaluate the quality of a point x:

$$\phi_1(x;\mu) \triangleq J(x) + \sum_{i\in\mathcal{E}} \mu_i |c_i(x)| + \sum_{i\in\mathcal{I}} \mu_i [c_i(x)]^-, \quad (8)$$

where $\mu_i > 0$ is the penalty parameter for each constraint i. The notation $[c_i]^- \triangleq \max\{0, -c_i\}$ is used to properly penalize inequality constraint violations.

Example 2 (Merit Function for the Toy Problem)**.** *Continuing Example 1, we plot the merit function for the toy MPCC problem (1) with different penalty parameters in Fig. 3. Evidently, the stationary points of the merit function align with the global minimum of Example 1 at the origin.*

Remark 3 (Merit Function)**.** *The ℓ_1 penalty function is a class of nonsmooth penalty functions that has demonstrated remarkable success in practical applications. It is exact in the sense that, given an appropriate penalty parameter and under mild constraint qualifications, local optimizers of the original problem are also minimizers of the merit function [19].*

We need to emphasize that, due to the lack of CQs in the MPCC problem, it is not fully clear whether minimization of the merit function (8) has the same type of guarantees as in the case where CQs hold. However, the intuition that the merit function balances objective reduction and constraint satisfaction still holds, and according to Example 2, the minimization of the merit function leads to the optimal solution, at least for the toy problem. We give a sufficient condition of local optimality in Proposition 9 and leave a precise investigation of the relationship between the merit function and the original problem in the case of MPCC for future work.

A difference of our merit function from the usual ℓ_1 penalty is that we maintain a separate penalty parameter μ_i for each constraint. This individualized penalization strategy often leads to superior convergence in practice.

Convex subproblem. To minimize the merit function (8), we adopt a sequential optimization strategy that solves a series of subproblems by approximating a local model of the merit function in the vicinity of the current iteration. Formally, our subproblem seeks to minimize the following quadratic model:

$$\begin{aligned} \min_{p_k} q_{\mu,k}(p_k) := \; & J_k + \nabla J_k^\top p_k + \frac{1}{2} p_k^\top \nabla^2_{xx} J_k \, p_k \\ & + \sum_{i\in\mathcal{E}} \mu_i |c_i(x_k) + \nabla c_i(x_k)^\top p_k| \\ & + \sum_{i\in\mathcal{I}} \mu_i [c_i(x_k) + \nabla c_i(x_k)^\top p_k]^-, \end{aligned} \quad (9)$$

where J_k, ∇J_k, and $\nabla^2_{xx} J_k$ represent the objective function's value, gradient, and Hessian at the current iterate x_k, respectively. The constraints c_i are linearized around x_k. In the subproblem (9), "p_k" is known as the *trial step*.

From (9), the readers see that our method diverges from the subproblem formulation in the classical SQP methods [23, 20, 30], in the sense that (9) refrains from computing second-order derivatives of the constraints and the Lagrangian. Instead, we linearize the constraints while maintaining a quadratic model of the objective function only. Our approach aligns with the concept of sequential convex programming discussed in [14, 13, 43, 39, 38], including the well-known robotics package TRAJOPT [51]. TRAJOPT implements sequential convex programming for trajectory optimization and convex collision detection. However, it lacks user-friendly APIs to include contact constraints and does not support weighted ℓ_1 merit functions or second-order corrections, nor does it provide convergence analysis. Beyond the SCP foundation, we aim for a high-performance implementation featuring carefully engineered modules and a generalized interface designed for broad contact-rich motion planning applications, while providing convergence guarantees in the contact-implicit context where CQs assumptions do not hold.

Leveraging the convexity of the objective function J and introducing the penalty term, the subproblem (9) is always convex and feasible. This offers several advantages. First, it reduces the computational burden associated with calculating second-order derivatives of constraints, which is particularly beneficial for problems with complex constraints (e.g., in robotics). Second, the convexity and guaranteed feasibility of the subproblem not only facilitate efficient solution of (9) using well-established convex optimization techniques but also enhance the robustness of the overall algorithm.

Remark 4 (Elastic Mode)**.** *In conventional sequential optimization methods, such as SQP, constraints are typically linearized directly from the original problem. However, when applied to MPCC, this approach inevitably leads to infeasible*

subproblems [22, 4]. To address this issue, some numerical solvers, like SNOPT*, implement an "Elastic Mode" where they add penalty variables to move constraints into the objective function upon detecting subproblem infeasibility. Our approach aligns with the motivation of the elastic mode in the sense that the merit function (8) is "elastic" from the very beginning because the constraints are penalized in the objective function. This strategy ensures our subproblems are always feasible by design.*

Globalization. The only issue with the subproblem (9) is that it may be unbounded from below (i.e., the optimal value is $-\infty$). To avoid this issue, we introduce the trust-region approach following [46]. Specifically, we constrain the ℓ_∞ norm of the trial step p_k. Formally, the trust-region subproblem with ℓ_∞ norm constraint becomes:

$$\begin{aligned}
\min_{p_k,v,w,t}\ & J_k + \nabla J_k^\mathsf{T} p_k + \frac{1}{2} p_k^\mathsf{T} \nabla^2_{xx} J_k p_k \\
& + \mu \sum_{i\in\mathcal{E}} (v_i + w_i) + \mu \sum_{i\in\mathcal{I}} t_i && \text{(10a)}\\
\text{subject to } & \nabla c_i(x_k)^\mathsf{T} p_k + c_i(x_k) = v_i - w_i,\ i \in \mathcal{E} && \text{(10b)}\\
& \nabla c_i(x_k)^\mathsf{T} p_k + c_i(x_k) \geq -t_i,\ i \in \mathcal{I} && \text{(10c)}\\
& v, w, t \geq 0 && \text{(10d)}\\
& \|p_k\|_\infty \leq \Delta_k, && \text{(10e)}
\end{aligned}$$

where (10e) represents the ℓ_∞ norm constraint with trust-region radius Δ_k. From (9) to (10), we have also introduced slack variables v, w, t to reformulate the nonsmooth ℓ_1 objective as smooth constraints. Evidently, the subproblem (10) is a standard convex QP.

Essentially, the trust region constraint (10e) acts as a filter on how much we trust the local model (quadratic objective and linearized constraints) to approximate the merit function. The ℓ_∞ norm only limits the maximum step length in the trial step, avoiding the issues related to ill-conditioning that can occur in line search methods.

To determine whether a trial step should be accepted and whether the trust-region radius should be adjusted, we monitor the ratio of actual decrease to predicted decrease:

$$\rho_k \triangleq \frac{\text{ared}_k}{\text{pred}_k} = \frac{\phi_1(x_k;\mu) - \phi_1(x_k + p_k;\mu)}{q_{\mu,k}(0) - q_{\mu,k}(p_k)}. \tag{11}$$

This ratio ρ_k serves as a key indicator of the quality of the trial step. As ρ_k approaches 1, it signifies that the trial step is increasingly satisfactory, indicating the local quadratic model at this step accurately describes the local behavior of the true merit function within the trust region radius. When the actual reduction is negative or the reduction ratio is very low, it indicates that the local model is highly inaccurate. In such cases, we typically shrink the trust region radius. To avoid continuous shrinking of the trust region and the Maratos effect, we introduce a second order correction step. The essence of this correction is to replace the linear approximation of $c(x_k + p)$ in subproblem (10) with a second-order model:

$$c(x_k + p) \approx c(x_k) + \nabla c(x_k)^T p + \frac{1}{2} p^T \nabla^2_{xx} c(x_k) p. \tag{12}$$

Suppose we have calculated an unsatisfactory trial step p_k, and the correction step to be determined is not far from p_k. We approximate the second-order term $p^T \nabla^2_{xx} c(x_k) p$ utilizing the value $c(x_k + p_k)$ computed at the trial step:

$$\begin{aligned}
p^T \nabla^2_{xx} c(x_k) p &\approx p_k^T \nabla^2_{xx} c(x_k) p_k \\
&\approx c(x_k + p_k) - c(x_k) - \nabla c(x_k)^\top p_k. \qquad \text{(13)}
\end{aligned}$$

From (13), we use $c(x_k + p) = \nabla c(x_k)^\top p + (c(x_k + p_k) - \nabla c(x_k)^\top p_k)$ to replace the first-order approximation in (10b) and (10c) to calculate a correction step $\bar{p}_k$. This approach does not require explicit second-order information of the constraints, thereby maintaining computational efficiency. Furthermore, it preserves the convex QP structure of the subproblem. In practice, this second-order correction has proven highly effective in correcting inaccuracies of the linearized model, leading to more efficient reductions in the merit function.

Convergence analysis. We now provide convergence guarantees of CRISP to stationary points of the merit function (8). We first recall the definition of the directional derivative.

Definition 5 (Directional Derivative)**.** *The directional derivative of a function $f\colon \mathbb{R}^n \to \mathbb{R}$ in the direction p is:*

$$D(f;p) = \lim_{t\to 0} \frac{f(x + tp) - f(x)}{t}. \tag{14}$$

The stationary point of the nonsmooth merit function is:

Definition 6 (Stationary Point)**.** *A point $\bar{x}$ is a stationary point of the merit function $\phi_1(x,\mu)$ in (8) if its directional derivatives are nonnegative along all directions p, i.e.,*

$$D(\phi_1(\bar{x};\mu);p) \geq 0. \tag{15}$$

We then make mild assumptions about the motion planning problems under consideration.

Assumption 7 (Convexity)**.** *In problem (7), the objective function $J(x)$ is convex and continuous differentiable; c_i is differentiable and ∇c_i is Lipschitz continuous for all $i \in \mathcal{E}$ and $i \in \mathcal{I}$.*

We are ready to present the main result.

Theorem 8 (Convergence)**.** *Under Assumption 7, suppose:*

1) *The sequence of convex subproblems (10) converges, i.e., $x_k \to x^\star$ for some point $x^\star$*
2) *The trust region radius remains bounded above zero, i.e., there exists $\Delta_{\min} > 0$ such that $\Delta_k \geq \Delta_{\min}$ for all k.*

Then, $x^\star$ is a stationary point of the merit function (8).

Proof: See Appendix A. ■

While the nonsmoothness of the merit function makes it challenging to verify whether its stationary points are local minima, a correspondence between the local minima of the merit function and the original problem can be established.

Proposition 9 (Local Optimality)**.** *Let $x^\star$ be a local minimizer of the merit function $\phi_1(x;\mu)$ in (8). If $x^\star$ is feasible for the*

original problem (7)*, that is,*

$$c_i(x^\star) = 0, \quad i \in \mathcal{E} \tag{16}$$
$$c_i(x^\star) \geq 0. \quad i \in \mathcal{I} \tag{17}$$

Then $x^\star$ is also a local minimizer of the original problem (7)*.*

Proof: Since $x^\star$ is a local minimizer of $\phi_1(x;\mu)$, by definition, there exist $r > 0 \;\; s.t. \;\; \phi_1(x^\star;\mu) \leq \phi_1(x;\mu), \;\; \forall ||x - x^\star|| < r$. Define the set $||x - x^\star|| < r$ as $B_r(x^\star)$ and let the feasible set of the original problem (7) be Ω. Then, we have

$$\phi_1(x^\star;\mu) \leq \phi_1(x;\mu). \;\; \forall x \in B_r(x^\star) \cap \Omega$$

On the other hand, we have $\phi_1(x;\mu) = J(x) \; \forall x \in \Omega$. Thus, evaluating the objective function of the original problem, we have:

$$J(x^\star) \leq J(x), \quad \forall x \in B_r(x^\star) \cap \Omega, \quad x^\star \in \Omega$$

which is the definition of a local minimizer. ■

Combining Theorem 8 and Proposition 9, we conclude that if (a) CRISP converges to a stationary point $x^\star$ of the merit function, (b) $x^\star$ is a local minimizer of the merit function, and (c) $x^\star$ is feasible for the original problem (7), then $x^\star$ is also a local minimizer of (7). In other words, when CRISP computes a stationary point $x^\star$ of the merit function (which often occurs and can be certified) that is also locally optimal, the only way that $x^\star$ might fail to be a local minimizer of the original problem (7) is that $x^\star$ is infeasible for (7). We note that while this statement is theoretically useful, due to the nonsmoothness of the merit function, certifying local optimality of $x^\star$ for the merit function is challenging. We leave this for future work.

Final algorithm. Having introduced the key algorithmic components and the convergence analysis, we formally present CRISP in Algorithm 1. The algorithm initiates with a common penalty variable μ_0 for all constraints. Guided by the checkable convergence conditions outlined in Theorem 8, we solve a series of trust-region subproblems (convex QPs) in the inner iterations. This process continues until the conditions in Theorem 8 are satisfied, indicating convergence to a stationary point of the current merit function. In the outer iterations, we examine the constraint violation against a threshold ϵ_c. For constraint violations exceeding this threshold, we increase their corresponding penalty parameters. The default values of all hyperparameters in Algorithm 1 are summarized in Table I.

III. CRISP: Implementation

User interface. CRISP follows the general nonlinear programming problem formulation in (7) where the user defines the objective function and constraints. Our implementation builds upon the fundamental data types in Eigen3 (vectors and matrices), wrapped with CPPAD [7] and CPPAD Code Generation (CG) [35]. This architecture allows users to define problems in an Eigen-style syntax, specifying objectives and constraints (both equality and inequality) while leveraging CPPAD for automatic derivative computation. The solver utilizes CPPAD to automatically obtain derivative information, while CG is employed to generate and store an auto-diff library. This library can be dynamically loaded for efficient evaluation of gradients and Hessians at specified points, significantly reducing computational overhead in subsequent solves. Besides, our implementation supports both parameterized and non-parameterized function definitions, allowing real-time modification of problem parameters (e.g., reference, terminal, or initial states) without the need to regenerate the library. It is noteworthy to point that

QP solver. The key subroutine in CRISP is to solve the convex trust-region QPs. After careful comparison and experimentation with several QP solvers in CRISP, including OSQP [54], PROXQP [6], qpOASE [18], and PIQP [52], we integrate PIQP, an interior-point method-based QP solver with embedded sparse matrix operations, which strikes an excellent balance of accuracy and real-time performance. To optimize the construction of the QP subproblems (10), we perform memory-level operations for extracting, copying, and concatenating large-scale sparse matrices, bypassing the Eigen3 higher-level interfaces. This approach enhances the efficiency of sparse matrix operations that are crucial for complex robot motion planning problems.

To enhance usability, we also implement a Python interface using Pybind11. This interface allows users to adjust problem parameters and solver hyperparameters, solve and re-solve problems, and extract results within a Python environment, providing flexibility for downstream applications.

Remark 10 (First-order QP Solver)**.** *There are growing interests in robotics to leverage first-order QP solvers (e.g., building upon ADMM [55] and PDHG [37]) for large-scale motion planning. While first-order QP solvers are typically more scalable due to cheap per-iteration cost, recent work [32] has shown that insufficient solution quality caused by first-order QP solvers may hurt motion planning. For this reason,* CRISP *chose the interior-point solver PIQP to make sure the inner QPs are solved to sufficient accuracy. However, we do note that* CRISP *is built in a modular way such that users can switch between different QP solvers.*

IV. CRISP: Benchmark

In this section, we present a comprehensive benchmark study to evaluate the performance of our proposed solver, CRISP, against several state-of-the-art optimization algorithms. As illustrated in Fig. 4, we have selected six representative contact-rich planning problems, each involving intricate interactions of normal (support) forces and friction forces. These problems showcase the diverse challenges encountered in practical applications where contact dynamics play a crucial role. All optimization problems are derived using the contact-implicit formulation and the detailed derivations are presented in Appendix B. The optimization horizon for push T is 50 with time discretization $dt = 0.05$ seconds, and all other examples are optimized over 200 steps with $dt = 0.02$ seconds. All experiments were conducted on a laptop equipped with a 13th Gen Intel(R) Core(TM) i9-13950HX processor.

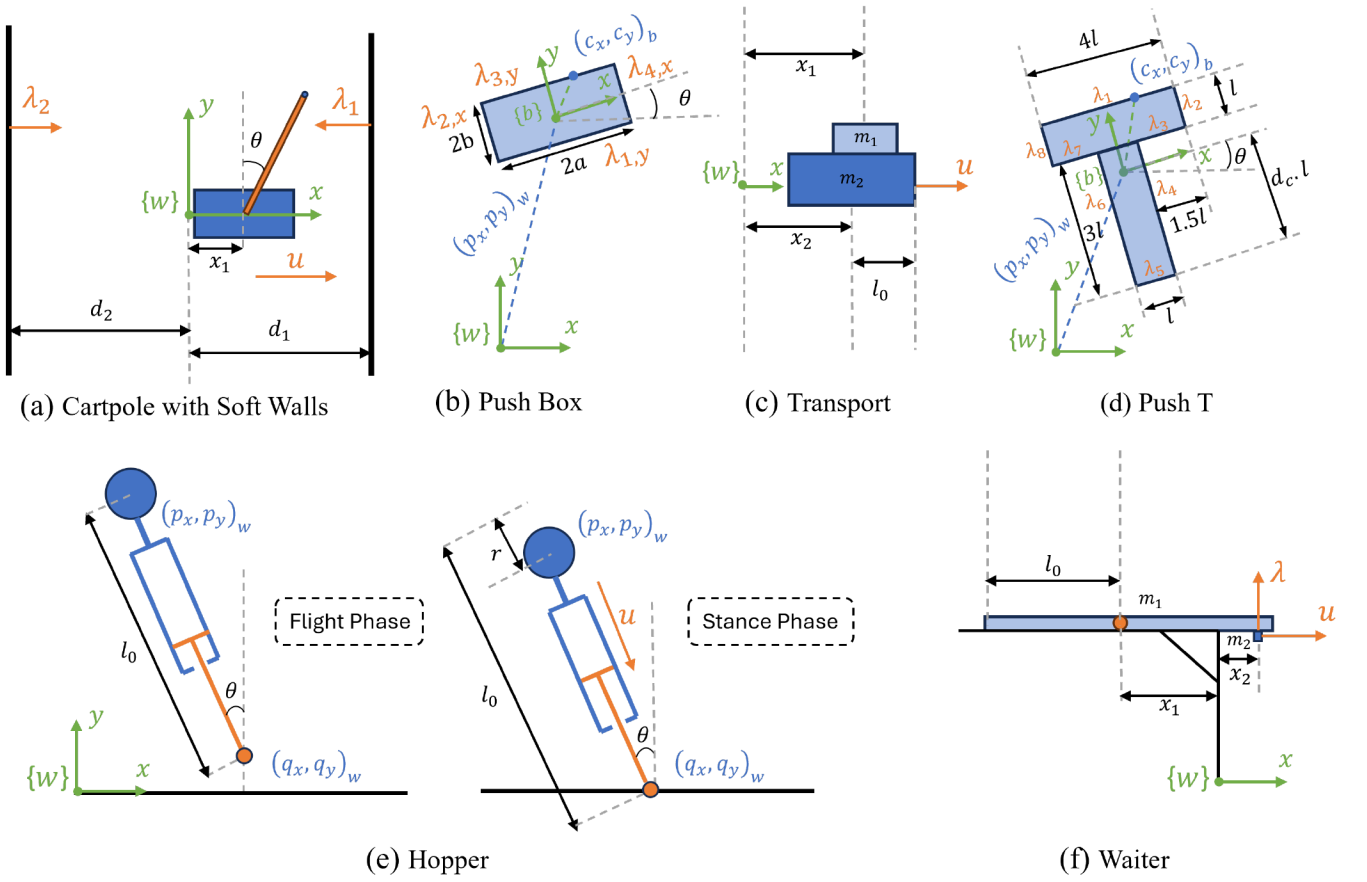


Fig. 4: Schematic overview of the contact-implicit motion planning tasks considered in the experiments. Each task poses unique challenges in contact sequencing, force distribution, and modeling of the multi-modal dynamics.

A. Benchmark Solvers

In the first part (§IV-C), we compare CRISP against three state-of-the-art algorithms. The first is SNOPT [24], a sparse nonlinear optimizer based on the SQP method. The second is IPOPT [57], an interior point optimizer designed for large-scale nonlinear programming. Finally, we include PROXNLP [29], a primal-dual augmented Lagrangian algorithm designed by roboticists. in subsection §IV-D, we present additional experiments to further demonstrate CRISP's robustness and superior performance across a broader range of contact-rich motion planning scenarios. Specifically, (*i*) For the push T task, we reformulate the problem as a mixed-integer quadratically constrained problem (MIQCP), which we solve using Gurobi [27]. (*ii*) We evaluate the LCP solver LCQpow [28] on two examples with linear dynamics and linear complementarity constraints. (*iii*) We validate CRISP in real-world Push T with a Franka Panda robot arm.

B. Benchmark Problems

For each problem depicted in Fig. 4, we provide a brief description of the task along with the initial conditions and initial guesses supplied to the solvers. The detailed formulations for each example, including objective functions, dynamics, contact constraints, and other general constraints, are provided in Appendix B. Furthermore, we have open-sourced all these problems as examples in CRISP.

1) Cartpole with soft walls: As shown in Fig. 4(a), this problem involves trajectory optimization of a cartpole system constrained between two walls, with the objective of swinging up the pole to an upright stance. The walls are considered "soft", providing unidirectional forces analogous to springs. Depending on the initial conditions, the optimal trajectory may require the pole to make or break contact with the walls, thereby leveraging contact forces to accomplish the task. x_1 and θ represent the cartpole states; u is the horizontal active force applied on the cart, while λ_1 and λ_2 are the passive contact forces generated when the pole's end makes contact with a wall. These contact forces are proportional to the wall displacement, effectively simulating the soft contact forces. Detailed contact-implicit formulation can be found Appendix B-A. We consider two distinct scenarios in the comparison based on the cart's initial position: centered between the walls and in proximity to one wall. For each scenario, we select five different initial conditions, with varying initial pole angles and velocities. For each initial condition, we provide two types of initial guesses: passive trajectories rolled out under zero control input, and these trajectories

Algorithm 1: CRISP

Input: Initial guess x_0

Data: $k_{\max}$, Δ_0, $\Delta_{\max}$, μ_0, $\mu_{\max}$, η_{low}, η_{high}, γ_{shrink}, γ_{expand}, ϵ_c, ϵ_p, ϵ_r

1 Initialize $k \leftarrow 0$, $\mu \leftarrow \mu_0$, $\Delta_k \leftarrow \Delta_0$;
2 **while** $k < k_{\max}$ **do**
3 Evaluate f_k, ∇f_k, $\nabla^2_{xx} f_k$, c_k, and ∇c_k at x_k;
4 Construct and solve subproblem (10) to obtain p_k;
5 Compute reduction ratio ρ_k;
6 // Second Order Correction;
7 **if** $\text{ared}_k < 0$ **then**
8 From (13), calculate Second Order Correction Step $\bar{p}_k$;
9 $p_k \leftarrow \bar{p}_k$;
10 Re-compute reduction ratio ρ_k;
11 **if** $\text{ared}_k < 0$ **then**
12 Abandon the Step;
13 $\Delta_{k+1} \leftarrow \gamma_{\text{shrink}} \Delta_k$;
14 Go to Line 28;
15 **end**
16 **end**
17 // Trust Region Update;
18 **if** $\rho_k < \eta_{low}$ **then**
19 $\Delta_{k+1} \leftarrow \gamma_{\text{shrink}} \Delta_k$;
20 $x_{k+1} \leftarrow x_k + p_k$;
21 **else if** $\rho_k > \eta_{high}$ *and* $\|p_k\|_\infty = \Delta_k$ **then**
22 $\Delta_{k+1} \leftarrow \min\{\gamma_{\text{expand}} \Delta_k, \Delta_{\max}\}$;
23 $x_{k+1} \leftarrow x_k + p_k$;
24 **else**
25 $\Delta_{k+1} \leftarrow \Delta_k$;
26 $x_{k+1} \leftarrow x_k + p_k$;
27 **end**
28 // Check Convergence of Merit Function;
29 **if** $\Delta_{k+1} < \epsilon_r$ ***or*** $\|p_k\|_\infty < \epsilon_p$ **then**
30 Merit Function Converged;
31 Check Overall Convergence;
32 **if** *max constraint violation* $< \epsilon_c$ **then**
33 **return** $x^* \leftarrow x_k$;
34 Optimization Successful;
35 **else**
36 **for** $i \in \mathcal{E}$ *with* $|c_i(x_k)| \geq \epsilon_c$
37 ***and*** $i \in \mathcal{I}$ *with* $[c_i(x_k)]^- \geq \epsilon_c$ **do**
38 $\mu_i \leftarrow \min(10\mu_i, \mu_{\max})$;
39 **end**
40 **if** *any* $\mu_i > \mu_{\max}$ **then**
41 **return** *Failure*;
42 Penalty Max Out;
43 **end**
44 **end**
45 **end**
46 $k \leftarrow k + 1$;
47 **end**
48 Max Iterations Reached;
49 **return** *Failure;*

TABLE I: Definition of Hyperparameters in CRISP

Paramaters	Descriptions
k_{max}	max iteration numbers: 1000
Δ_0	initial trust region radius: 1
Δ_{max}	max trust region radius: 10
μ_0	initial penalty: 10
μ_{max}	max penalty: $1e^6$
η_{low}	reduction ratio lower bound: 0.25
η_{high}	reduction ratio upper bound: 0.75
γ_{shrink}	trust region shrink factor: 0.25
γ_{expand}	trust region expand factor: 2
ϵ_c	tolerance of constraint violation: $1e^6$
ϵ_p	tolerance of trial step norm: $1e^{-3}$
ϵ_r	tolerance of trust region radius: $1e^{-3}$

contaminated with random noise.

2) Push Box: The next benchmark problem is the classic push box problem, as illustrated in Fig. 4(b). In this task, our objective is to manipulate a planar box with dimensions $2a \times 2b$ to a series of specified target configurations on a table with friction. It involves multiple contact modes—four in this case, corresponding to different faces of the box. The solver must reason about both the sequence of contact positions and the appropriate contact forces to achieve the desired motion. The initial state of the system places the box at the origin, aligned with the positive x-axis. We provide an all-zero initial guess to the solvers, and the objective is to push the box to 18 different configurations. Each configuration is located 3 meters from the origin, with the angles distributed evenly in a clockwise direction, completing a full 360-degree circle. We refer to Appendix B-B for the contact-implicit formulation of different contact modes.
3) Transport: As shown in Fig. 4(c), the goal is to determine the active force u applied to the cart, under various initial poses and velocities, to transport the payload m_1 to a specified position without it falling off the cart m_2. This requires precise indirect manipulation of m_1 by controlling the friction force between m_1 and m_2 through u. We use an all-zero initial guess for different initial states. The cart is commanded to move from 3 meters to the origin, with different start and goal relative positions (leftmost, middle, rightmost) to the payload (e.g., precisely moving the payload from the leftmost to the rightmost part of the cart while reaching the absolute end position). The task becomes more challenging with different initial and final velocities, requiring alternating dragging motions of the cart to achieve the objective. Its MPCC formulation is provided in Appendix B-C.
4) Push T: The modeling techniques for the push T problem are similar to those used in the push box problem. However, it presents its own challenges due to the non-convex shapes of the "T" and the increase in contact modes ($\lambda_1 - \lambda_8$ as shown in Fig. 4(d)). This means we cannot simply construct complementarity constraints related to contact using four hyperplanes as we do with

the push box problem. We now need to establish complementarity conditions that involve contact points lying on certain line segments and exerting corresponding contact forces, which requires special modeling techniques. Details of the problem formulation is demonstrated in Appendix B-D. Together with the slack variables, the total state dimension is 29. The experiment setting is the same with Push Box.

5) Hopper: We model the system as a point mass m atop a massless leg with original length l_0, capable of radial contraction and thrust application. As shown in Fig. 4(e), the hopper exhibits two distinct phases: the "Flight Phase", where the leg length remains constant and motion approximates free fall, and the "Stance Phase", which initiates upon ground contact. During flight, we directly control the leg angular speed ($\dot{\theta} = u_1$). The stance phase, triggered by $p_y - l_0 \cos\theta \leq 0$, involves fixed-point rotation around the contact point. As the leg contracts (r increases within r_0), it can exert a unidirectional thrust $u = u_2$ along its radial axis. The transition back to flight occurs when $r = 0$. The task objective is to navigate the hopper's hybrid dynamics without pre-specified contact sequences. The hopper is released from a certain height at the origin, and must optimize its own contact sequence, stance phase contact forces, and flight phase angular velocities to jump and stop at $x = 2\,\text{m}$. This involves simultaneously managing multiple goals: exploring optimal contact sequences, controlling leg angle during flight, and managing thrust after landing to achieve the specified final position. The initial guess is a passive free-fall motion. As derived in Appendix B-E, we employ a contact-implicit formulation to unify the dynamics of both phases within a single framework, with carefully crafted complementarity constraints to govern the transitions between phases.
6) Waiter: The waiter problem, see Fig. 4(f), is also studied in [60] where it was modeled with linear complementarity constraints and optimized in a model-predictive control pipeline. Initially, the plate on the table has only a small overhang. The objective is to indirectly control the friction force with the plate by applying a normal force λ and a pulling force u to the plate m_1 via the pusher m_2. The goal is to gradually extract the plate until its center of mass x_1 aligns with the pusher's center x_2, stopping at the table's edge with matching velocities. This task requires consideration of the friction between the table and the plate, as well as preventing the plate from tipping over its leftmost contact point with the table (see Appendix B-F). We provide an all-zero initial guess to test the solvers' ability to optimize an informed contact trajectory.

C. Comparisons with Numerical Solvers

While achieving feasible trajectories is a primary goal for local solvers, this alone is insufficient in robotics applications. Dynamically feasible trajectories can often be trivially obtained (e.g., remaining stationary at the origin) but may lack practical value. Therefore, we argue that beyond mere feasibility, the ability to efficiently and robustly optimize informed control sequences for tracking objectives is crucial.

Performance metrics. We evaluate the solvers' performance based on three key metrics. First, we examine the solution success rate, which reflects the overall ability of each solver to converge to a *feasible* and *informed* solution from various settings (different initial/terminal conditions, different initial guesses). Second, we assess the quality of the solution through the objective values achieved and the tracking error. Finally, we measure the solving efficiency by considering the number of iterations required and the total solving time. These metrics serve to answer a fundamental scientific question:

Can these solvers robustly discover informed contact sequences and contact force profiles from scratch?

This inquiry is particularly relevant, as precise contact patterns are often unknown *a priori* in real-world applications.

Quantitative results. For the first four examples, we recorded and summarized the numerical performance of the benchmark solvers under different initial states and initial guesses, as presented in the Table II and Fig. 5. Solutions with constraints violation below 10^{-5}, translation error norm below 0.1, translation velocity error norm below 0.5, angular error below $\pi/6$, and angular velocity error below 0.1π rad/s as deemed successful. The objective function, as previously highlighted, primarily balances tracking error and control efforts. We analyze the performance of four solvers across the first four examples, focusing on median tracking error, average iteration count, and computation time. To better illustrate the performance and robustness of each solver under different initial guesses, we boxplot the distribution of constraint violations, objective values, and tracking errors in Fig. 5. These plots provide a comprehensive overview of solver behavior across all experiments, including both feasible and infeasible solutions, offering a complete picture of their performance characteristics.

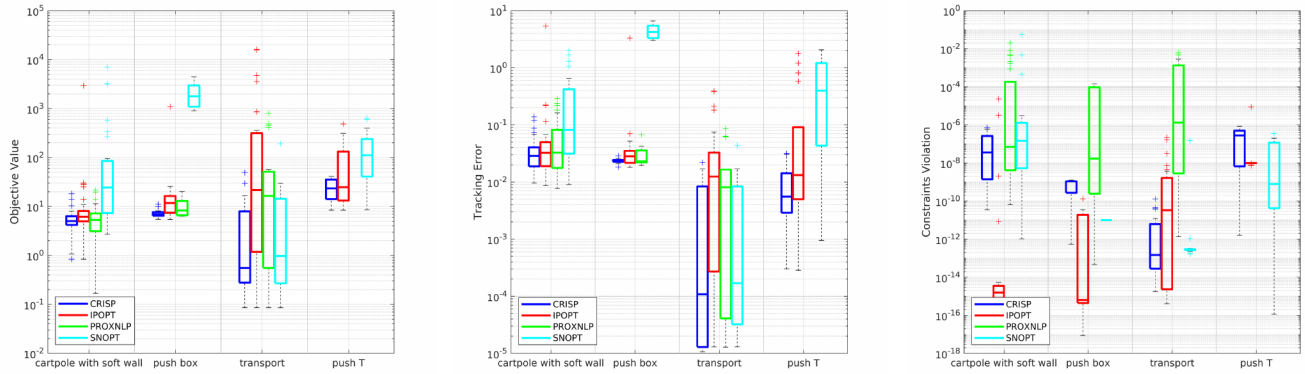
Our results demonstrate that CRISP *consistently produces feasible trajectories with the lowest tracking error and overall objective values across all tasks.*

The box plot underscores CRISP's remarkable robustness, showing consistently superior performance across varied initial states and guesses, with minimal outliers. While IPOPT generally performs well among the benchmark solvers, it occasionally produces feasible but impractical solutions. We've visualized these outlier cases, contrasting CRISP and IPOPT trajectories in Fig. 6. SNOPT, despite outperforming IPOPT in the transport example, struggles with efficiency due to high iteration requirements and exhibits unstable optimization results, notably failing to optimize the informed contact sequence in the push box scenario. These findings underscore the critical importance of solver stability. More visualization of the trajectories can be found in Appendix C.

Qualitative results. For the hopper and waiter examples, which involve more challenging contact reasoning, we focus our comparison on CRISP and IPOPT. The results are visualized in Fig. 7. In the hopper task, the objective is to jump and

Methods	Cartpole with Soft Walls					Push Box					Transport					Push T				
	Success Rate [%]	Tracking Error	Violation	Iterations	Time [s]	Success Rate [%]	Tracking Error	Violation	Iterations	Time [s]	Success Rate [%]	Tracking Error	Violation	Iterations	Time [s]	Success Rate [%]	Tracking Error	Violation	Iterations	Time [s]
SNOPT	63.33	0.08	2.0×10^{-3}	4552.50	12.97	0	4.25	1.0×10^{-11}	3303.00	25.22	**100**	1.7×10^{-4}	5.5×10^{-9}	5782.30	43.98	33.33	0.67	$\mathbf{6.4 \times 10^{-8}}$	2257.50	210.92
IPOPT	86.67	0.04	8.9×10^{-7}	**288.21**	**0.51**	94.44	0.03	$\mathbf{1.3 \times 10^{-11}}$	**407.69**	0.92	87.50	0.02	1.6×10^{-8}	404.96	1.36	72.22	0.26	4.9×10^{-7}	1560.20	39.49
PROXNLP	60	0.032	1.4×10^{-3}	716.25	140.92	66.67	0.03	3.6×10^{-5}	2000.00	1277.21	59.26	0.01	1.1×10^{-3}	361.92	1070.01	–	–	–	–	–
CRISP (ours)	**100**	**0.02**	$\mathbf{1.4 \times 10^{-7}}$	415.39	1.56	**100**	**0.02**	8.3×10^{-10}	420.69	**0.74**	**100**	$\mathbf{1.1 \times 10^{-4}}$	$\mathbf{1.1 \times 10^{-11}}$	**286.20**	**0.77**	**100**	**0.01**	8.7×10^{-7}	**311.35**	**3.54**

TABLE II: Comparison of CRISP with benchmark solvers across different tasks.



(a) Objective Value (b) Tracking Error (c) Constraints Violation

Fig. 5: Box plots of benchmark metrics on three tasks: cartpole with soft walls, push box, transport, and push T. The plots complement Table II to show the distribution of (a) objective value, (b) tracking error, and (c) constraint violation across multiple initial states and initial guesses. PROXNLP was unable to solve the Push T problem within a tractable amount of time due to its larger problem size, so its data is not applicable in the figure.

stop at a 2 m horizontal distance with zero height, while minimizing control effort. The waiter task requires maneuvering the plate's center of mass to the edge of the table without tipping, ensuring that both the plate and pusher have a final rightward velocity of 2 m/s. Both IPOPT and CRISP successfully optimized feasible trajectories, but CRISP significantly outperformed IPOPT in terms of solution quality for finding informed contact sequences. CRISP achieved remarkably low objective values of 5.45 and 8.68 for the hopper and waiter examples, respectively, with precise tracking performance. In contrast, IPOPT's solutions resulted in much higher objective values of 137.5 and 1855, failing to reach the desired terminal states and incurring substantially larger control efforts, which is evidently shown in Fig. 7.

Videos. On the project website, we provide animations of optimized trajectories computed by CRISP.

D. Additional Experiments

Additional Baselines on Push T. As an alternative formulation, we reformulated Push T as an MIQCP by associating each facet with a binary variable indicating the contact mode. Using Gurobi to solve this formulation, it failed to find solutions for planning horizons beyond 4 steps (within 10 minutes), making it incomparable with CRISP. The code to reproduce these experiments is open-sourced alongside CRISP.

LCP Solvers. The Transport and Waiter problems, characterized by linear dynamics and linear complementarity constraints, are naturally suited for a QP + LCP solver. We evaluated LCQpow [28] on these tasks. For the Transport problem, LCQpow achieves an average solve time of 251 s with an average tracking error of 0.4267, and 92.5% optimized trajectories are feasible. In comparison, as shown in Table II, CRISP significantly outperforms LCQpow with an average solve time of 0.77 s, an average tracking error of 1.1×10^{-4}. For the more challenging Waiter problem, LCQPow fails to generate any feasible solutions. The code for reproducing these results is available in our open-source repository.

Real-World Push T. To further validate the effectiveness of CRISP in delivering high-quality trajectories, we applied CRISP to a real robotic arm system for the push T task. In our validation experiments, we estimated the position of the T block during each perception cycle, using this as the new initial state for CRISP to solve them online. The solution from CRISP was then applied in an MPC feedback loop. In the physical experiments, we discretized the problem into 10 time steps with intervals of 0.05 s. Under this setup, the average solution time for CRISP was 80 ms, demonstrating robust and efficient performance in pushing the T block to the target position. A visualization of the process for different initial conditions is provided in Fig. 8.

V. RELATED WORKS

Motion planning through contact presents unique challenges due to the inherently discontinuous nature of contact inter-

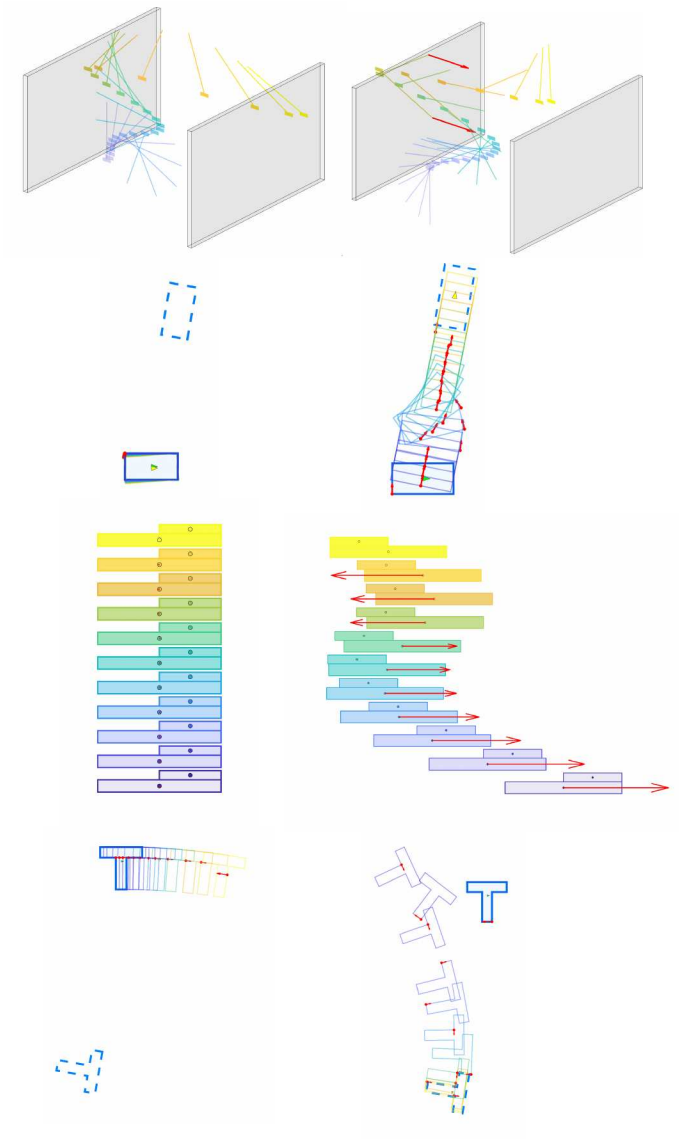


Fig. 6: Visualization of some outlier cases in IPOPT. IPOPT results (left column) versus our approach (right column). The color gradient represents the progression of time (from blue to yellow). We note that in the transport example (the third row), the initial state has the cargo at the rightmost end of the cart, with both moving left at 4 m/s. The terminal state requires the cart at the origin and the cargo at the leftmost end, both with a leftward velocity of 2 m/s. This scenario necessitates initial braking, followed by back-and-forth motion near the origin. Actually, this challenging case causes all other solvers to fail. While IPOPT produces a feasible but impractical solution (remaining stationary at the origin), our approach successfully optimizes the complex contact sequences.

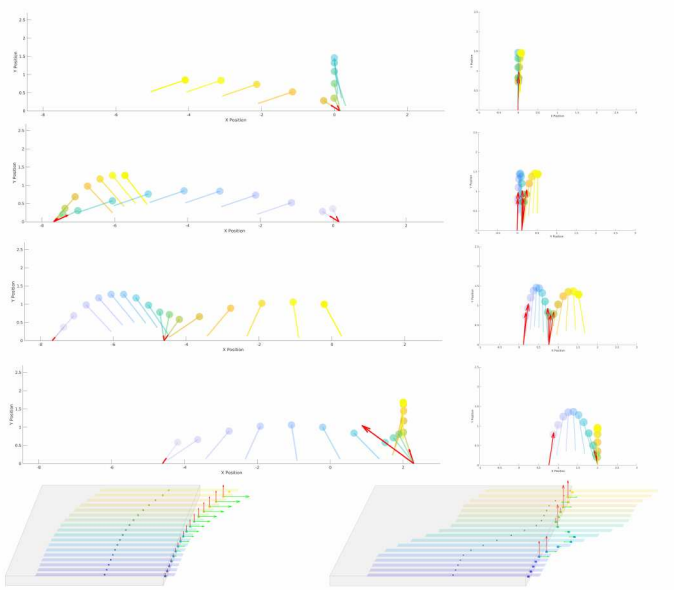


Fig. 7: Comparison of optimized trajectories for the hopper and waiter problems. The left and right columns show the results from IPOPT and CRISP respectively. In the hopper problem, we present four key frames (from top to bottom at 1 s, 2 s, 3 s, and 4 s) illustrating the hopper's trajectories and contact force. The yellow color indicates the current position of the hopper, while blue represents the past trail.

actions [58, 34]. The literature on modeling contact can be broadly divided into smooth and rigid methods.

Smooth contact model. The principle of modeling contact in a "smooth" manner involves approximating nonsmooth contact events into smooth and continuous functions relating contact forces to states. This approach often simulates effects similar to springs [9], dampers [41], or a combination of both [44, 45]. By doing so, it allows contact forces to be expressed as functions of the robot's states and seamlessly integrated into the overall dynamic functions, providing well-defined gradient information.

Rigid contact model: hybrid dynamics. Hybrid systems offer a robust framework for modeling systems that exhibit both continuous and discrete behaviors [25]. These systems are characterized by their ability to switch between different dynamic regimes, or modes, depending on the contact conditions. In the locomotion community, the control of switched systems often allows for instantaneous changes in velocity during contact events [17, 16, 1], while continuous dynamics govern the system at other times. This approach requires a predefined gait, which specifies a sequence of potential contact points [8, 11, 36].

Rigid contact model: implicit formulations. There are two mathematically equivalent approaches to implicitly encode the discrete nature of hybrid systems for switching between different continuous subsystems. One approach is through Mixed Integer Programming (MIP) [56], which introduces binary integer variables to act as switches for encoding

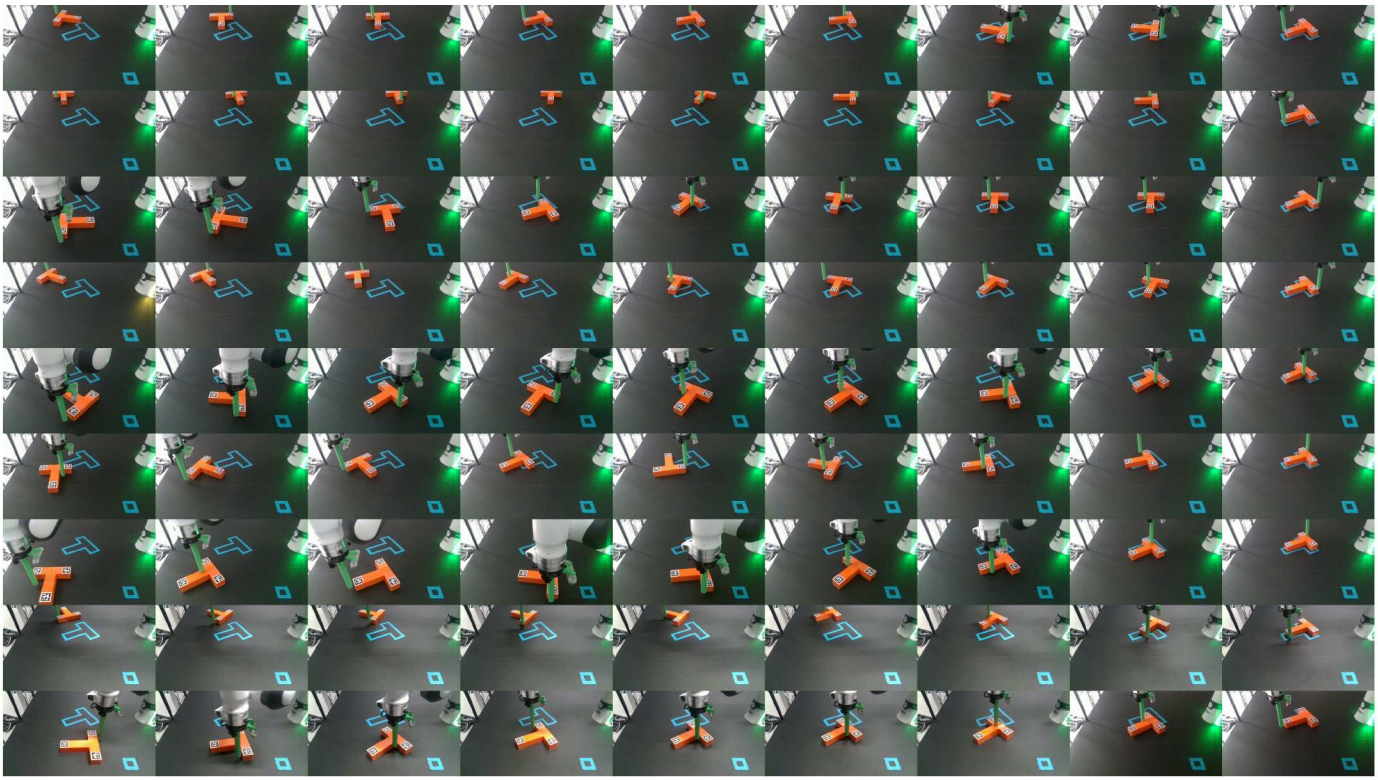


Fig. 8: Key frames visualization of the real-world Push T experiments from various initial configurations (each row). Videos of the results can be found on the project website.

contact events [12, 2, 3]. This method is straightforward in its implementation, yet optimizing these discrete variables is challenging for gradient-based NLP solvers and often requires specialized solvers, such as Gurobi [27]. Furthermore, the number of integer variables can significantly increase with the number of contact modes and the planning horizon, leading to computational intractability [53, 40]. On the other hand, the contact force and condition can be encoded through the introduction of complementarity constraints.

On the other hand, contact forces and conditions can be encoded through the introduction of complementarity constraints. Since [47], this approach has recently gained attention because it transforms the problem into a continuous NLP problem without compromising the discrete characteristics of contact. This transformation allows for the use of modern numerical optimization tools to solve the problem effectively [33, 60, 5]. However, the failure of CQs in this context can lead to significant difficulties in solving these problems [20, 22, 21]. CRISP aims to provide an efficient and robust solution by addressing the challenges associated with solving nonlinear contact problems that include general nonlinear complementarity constraints.

VI. Conclusion

We presented CRISP, a primal-only numerical solver for contact-implicit motion planning that is based on sequential convex optimization. We started by uncovering the geometric insights underpinning the difficulty of solving MPCCs arising from contact-implicit planning. That motivated us to design a primal-only algorithm where each trust-region subproblem is convex and feasible by construction. For the first time, we proved sufficient conditions on the algorithm's convergence to stationary points of the merit function. With a careful C++ implementation, we benchmarked CRISP against state-of-the-art solvers on six contact-implicit planning problems, demonstrating superior robustness and capability to generate entirely new contact sequences from scratch. We believe CRISP sets a new standard and our open-source benchmarks offer significant value to contact-rich motion planning—one of the most active and challenging areas in robotics research.

Limitations and future work. First, while CRISP accepts general nonlinear programming problem definitions applicable to various optimal control and motion planning problems, it requires further development to evolve into a comprehensive robotics optimization toolbox comparable to OCS2 [17] and CROCODDYL [42]. This development primarily involves integration with advanced dynamics libraries such as Pinocchio [10], which would enable CRISP to handle more complex dynamics and obtain their derivative information efficiently. Second, although our C++ implementation is highly efficient, its adaptivity for real-time tasks could be further increased. The acquisition of gradients and Hessian matrices could be parallelized, and matrix operations could achieve significant speedups if implemented on GPUs [31]. Additionally, while

our subproblems are already cheap as convex QPs, the overall framework's efficiency is still constrained by the speed of the underlying QP solver. As mentioned in Remark 10, we plan to test CRISP with scalable first-order QP solvers. On the theoretical side, a complete picture of the relationship between local solutions of the original problem and stationary points of the merit function, as well as how to check if a stationary point of the nonsmooth merit function (that CRISP finds) is locally optimal, warrants further investigation.

Acknowledgment

We thank Michael Posa and Zac Manchester for insightful discussions about contact-rich motion planning.

References

[1] Sequential linear quadratic optimal control for nonlinear switched systems. *IFAC-PapersOnLine*, 50(1):1463–1469, 2017. ISSN 2405-8963. 20th IFAC World Congress. 11

[2] Bernardo Aceituno-Cabezas, Hongkai Dai, José Cappelletto, Juan C. Grieco, and Gerardo Fernández-López. A mixed-integer convex optimization framework for robust multilegged robot locomotion planning over challenging terrain. In *2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 4467–4472, 2017. 12

[3] Bernardo Aceituno-Cabezas, Carlos Mastalli, Hongkai Dai, Michele Focchi, Andreea Radulescu, Darwin G. Caldwell, José Cappelletto, Juan C. Grieco, Gerardo Fernández-López, and Claudio Semini. Simultaneous contact, gait, and motion planning for robust multilegged locomotion via mixed-integer convex optimization. *IEEE Robotics and Automation Letters*, 3(3):2531–2538, 2018. 12

[4] Mihai Anitescu. On using the elastic mode in nonlinear programming approaches to mathematical programs with complementarity constraints. *SIAM Journal on Optimization*, 15(4):1203–1236, 2005. 5

[5] Alp Aydinoglu and Michael Posa. Real-time multi-contact model predictive control via admm. In *2022 International Conference on Robotics and Automation (ICRA)*, pages 3414–3421. IEEE, 2022. 3, 12

[6] Antoine Bambade, Sarah ElKazdadi, Adrien Taylor, and Justin Carpentier. PROX-QP: Yet another quadratic programming solver for robotics and beyond. In *Robotics: Science and Systems*, 2022. 6

[7] Bradley M. Bell. CppAD: A package for C++ algorithmic differentiation (2024). https://github.com/coin-or/CppAD. 6

[8] Gerardo Bledt, Matthew J. Powell, Benjamin Katz, Jared Di Carlo, Patrick M. Wensing, and Sangbae Kim. Mit cheetah 3: Design and control of a robust, dynamic quadruped robot. In *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 2245–2252, 2018. 11

[9] R. Blickhan. The spring-mass model for running and hopping. *Journal of Biomechanics*, 22(11):1217–1227, 1989. ISSN 0021-9290. 11

[10] Justin Carpentier, Guilhem Saurel, Gabriele Buondonno, Joseph Mirabel, Florent Lamiraux, Olivier Stasse, and Nicolas Mansard. The Pinocchio C++ library – a fast and flexible implementation of rigid body dynamics algorithms and their analytical derivatives. In *IEEE International Symposium on System Integrations (SII)*, 2019. 12

[11] Xianyi Cheng, Eric Huang, Yifan Hou, and Matthew T. Mason. Contact mode guided motion planning for quasidynamic dexterous manipulation in 3d. In *2022 International Conference on Robotics and Automation (ICRA)*, pages 2730–2736, 2022. 11

[12] Robin Deits and Russ Tedrake. Footstep planning on uneven terrain with mixed-integer convex optimization. In *2014 IEEE-RAS International Conference on Humanoid Robots*, pages 279–286, 2014. 12

[13] Moritz Diehl and Florian Messerer. Local convergence of generalized gauss-newton and sequential convex programming. In *2019 IEEE 58th Conference on Decision and Control (CDC)*, pages 3942–3947, 2019. 4

[14] Quoc Tran Dinh and Moritz Diehl. Local convergence of sequential convex programming for nonconvex optimization. In *Recent Advances in Optimization and its Applications in Engineering*, pages 93–102. Berlin, Germany: Springer, 2010. 4

[15] Francisco Facchinei, Houyuan Jiang, and Liqun Qi. A smoothing method for mathematical programs with equilibrium constraints. *Mathematical programming*, 85(1): 107, 1999. 2

[16] Farbod Farshidian, Edo Jelavic, Asutosh Satapathy, Markus Giftthaler, and Jonas Buchli. Real-time motion planning of legged robots: A model predictive control approach. In *2017 IEEE-RAS 17th International Conference on Humanoid Robotics (Humanoids)*, pages 577–584, 2017. 11

[17] Farbod Farshidian et al. OCS2: An open source library for optimal control of switched systems. [Online]. Available: https://github.com/leggedrobotics/ocs2. 11, 12

[18] H.J. Ferreau, C. Kirches, A. Potschka, H.G. Bock, and M. Diehl. qpOASES: A parametric active-set algorithm for quadratic programming. *Mathematical Programming Computation*, 6(4):327–363, 2014. 6

[19] R. Fletcher. *Penalty Functions*, pages 87–114. Berlin, Germany: Springer, 1983. 4

[20] Roger Fletcher. *Practical methods of optimization*. Hoboken, NJ, USA: Wiley, 2013. 3, 4, 12

[21] Roger Fletcher and Sven Leyffer. Solving mathematical programs with complementarity constraints as nonlinear programs. *Optimization Methods and Software*, 19(1): 15–40, 2004. 3, 12

[22] Roger Fletcher, Sven Leyffer, Danny Ralph, and Stefan Scholtes. Local convergence of sqp methods for mathematical programs with equilibrium constraints. *SIAM*

Journal on Optimization, 17(1):259–286, 2006. 3, 5, 12

[23] Philip E. Gill and Elizabeth Wong. Sequential quadratic programming methods. In *Mixed Integer Nonlinear Programming*, pages 147–224. New York, NY, USA: Springer, 2012. 4

[24] Philip E. Gill, Walter Murray, and Michael A. Saunders. SNOPT: An sqp algorithm for large-scale constrained optimization. *SIAM Journal on Optimization*, 12(4):979–1006, 2002. 2, 7

[25] Rafal Goebel, Ricardo G. Sanfelice, and Andrew R. Teel. Hybrid dynamical systems. *IEEE Control Systems Magazine*, 29(2):28–93, 2009. 11

[26] Bernhard Paus Graesdal, Shao Yuan Chew Chia, Tobia Marcucci, Savva Morozov, Alexandre Amice, Pablo A. Parrilo, and Russ Tedrake. Towards tight convex relaxations for contact-rich manipulation, 2024. URL https://arxiv.org/abs/2402.10312. 18

[27] Gurobi Optimization, LLC. Gurobi Optimizer Reference Manual, 2024. URL https://www.gurobi.com. 7, 12

[28] J. Hall, A. Nurkanović, F. Messerer, and M. Diehl. LCQPow: a solver for linear complementarity quadratic programs. *Math. Prog. Comput.*, 2024. 3, 7, 10

[29] Wilson Jallet, Antoine Bambade, Nicolas Mansard, and Justin Carpentier. Constrained differential dynamic programming: A primal-dual augmented lagrangian approach. In *2022 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 13371–13378, 2022. 7

[30] Armand Jordana, Sébastien Kleff, Avadesh Meduri, Justin Carpentier, Nicolas Mansard, and Ludovic Righetti. Stagewise implementations of sequential quadratic programming for model-predictive control. *Preprint*, 2023. 4

[31] Shucheng Kang, Xiaoyang Xu, Jay Sarva, Ling Liang, and Heng Yang. Fast and certifiable trajectory optimization. *arXiv preprint arXiv:2406.05846*, 2024. 12

[32] Charles Khazoom, Seungwoo Hong, Matthew Chignoli, Elijah Stanger-Jones, and Sangbae Kim. Tailoring solution accuracy for fast whole-body model predictive control of legged robots. *IEEE Robotics and Automation Letters*, 2024. 6

[33] Simon Le Cleac'h, Taylor A Howell, Shuo Yang, Chi-Yen Lee, John Zhang, Arun Bishop, Mac Schwager, and Zachary Manchester. Fast contact-implicit model predictive control. *IEEE Transactions on Robotics*, 2024. 2, 3, 12

[34] Quentin Le Lidec, Wilson Jallet, Louis Montaut, Ivan Laptev, Cordelia Schmid, and Justin Carpentier. Contact models in robotics: A comparative analysis. *IEEE Transactions on Robotics*, 40:3716–3733, 2024. doi: 10.1109/TRO.2024.3434208. 11

[35] Joao Rui Leal. CppADCodeGen: About source code generation for automatic differentiation using operator overloading (2024). https://github.com/joaoleal/CppADCodeGen. 6

[36] Gabriel A. D. Lopes, Bart Kersbergen, Ton J. J. van den Boom, Bart De Schutter, and Robert Babuška. Modeling and control of legged locomotion via switching max-plus models. *IEEE Transactions on Robotics*, 30(3):652–665, 2014. 11

[37] Haihao Lu and Jinwen Yang. A practical and optimal first-order method for large-scale convex quadratic programming. *arXiv preprint arXiv:2311.07710*, 2023. 6

[38] Yuanqi Mao, Michael Szmuk, and Behçet Açıkmeşe. Successive convexification of non-convex optimal control problems and its convergence properties. In *2016 IEEE 55th Conference on Decision and Control (CDC)*, pages 3636–3641, 2016. doi: 10.1109/CDC.2016.7798816. 4

[39] Yuanqi Mao, Michael Szmuk, Xiangru Xu, and Behçet Açikmese. Successive convexification: A superlinearly convergent algorithm for non-convex optimal control problems. *arXiv preprint arXiv:1804.06539*, 2018. 4

[40] Tobia Marcucci and Russ Tedrake. Warm start of mixed-integer programs for model predictive control of hybrid systems. *IEEE Transactions on Automatic Control*, 66 (6):2433–2448, 2021. 12

[41] D.W. Marhefka and D.E. Orin. A compliant contact model with nonlinear damping for simulation of robotic systems. *IEEE Transactions on Systems, Man, and Cybernetics - Part A: Systems and Humans*, 29(6):566–572, 1999. 11

[42] Carlos Mastalli, Rohan Budhiraja, Wolfgang Merkt, Guilhem Saurel, Bilal Hammoud, Maximilien Naveau, Justin Carpentier, Ludovic Righetti, Sethu Vijayakumar, and Nicolas Mansard. Crocoddyl: An Efficient and Versatile Framework for Multi-Contact Optimal Control. In *IEEE International Conference on Robotics and Automation (ICRA)*, 2020. 12

[43] Florian Messerer and Moritz Diehl. Determining the exact local convergence rate of sequential convex programming. In *2020 European Control Conference (ECC)*, pages 1280–1285, 2020. 4

[44] Michael Neunert, Farbod Farshidian, Alexander W. Winkler, and Jonas Buchli. Trajectory optimization through contacts and automatic gait discovery for quadrupeds. *IEEE Robotics and Automation Letters*, 2(3):1502–1509, 2017. 11

[45] Michael Neunert, Markus Stäuble, Markus Giftthaler, Carmine D. Bellicoso, Jan Carius, Christian Gehring, Marco Hutter, and Jonas Buchli. Whole-body nonlinear model predictive control through contacts for quadrupeds. *IEEE Robotics and Automation Letters*, 3 (3):1458–1465, 2018. 11

[46] Jorge Nocedal and Stephen J Wright. *Numerical optimization*. New York, NY, USA: Springer, 2006. 2, 4, 5

[47] Michael Posa, Cecilia Cantu, and Russ Tedrake. A direct method for trajectory optimization of rigid bodies through contact. *The International Journal of Robotics Research*, 33(1):69–81, 2014. 1, 12

[48] Holger Scheel and Stefan Scholtes. Mathematical programs with complementarity constraints: Stationarity,

optimality, and sensitivity. *Mathematics of Operations Research*, 25(1):1–22, 2000. 2

[49] Stefan Scholtes. Convergence properties of a regularization scheme for mathematical programs with complementarity constraints. *SIAM Journal on Optimization*, 11 (4):918–936, 2001. 2

[50] Stefan Scholtes and Michael Stöhr. How stringent is the linear independence assumption for mathematical programs with complementarity constraints? *Mathematics of Operations Research*, 26(4):851–863, 2001. 2

[51] John Schulman, Jonathan Ho, Alex X Lee, Ibrahim Awwal, Henry Bradlow, and Pieter Abbeel. Finding locally optimal, collision-free trajectories with sequential convex optimization. In *Robotics: science and systems*, volume 9, pages 1–10. Berlin, Germany, 2013. 4

[52] Roland Schwan, Yuning Jiang, Daniel Kuhn, and Colin N. Jones. PIQP: A proximal interior-point quadratic programming solver. In *2023 62nd IEEE Conference on Decision and Control (CDC)*, pages 1088–1093, 2023. 6

[53] Yuki Shirai, Devesh K. Jha, Arvind U. Raghunathan, and Diego Romeres. Chance-constrained optimization for contact-rich systems using mixed integer programming. *Nonlinear Analysis: Hybrid Systems*, 52:101466, 2024. ISSN 1751-570X. 12

[54] B. Stellato, G. Banjac, P. Goulart, A. Bemporad, and S. Boyd. OSQP: an operator splitting solver for quadratic programs. *Mathematical Programming Computation*, 12 (4):637–672, 2020. 6

[55] Bartolomeo Stellato, Goran Banjac, Paul Goulart, Alberto Bemporad, and Stephen Boyd. OSQP: An operator splitting solver for quadratic programs. *Mathematical Programming Computation*, 12(4):637–672, 2020. 6

[56] Juan Pablo Vielma. Mixed integer linear programming formulation techniques. *SIAM Review*, 57(1):3–57, 2015. 11

[57] Andreas Wächter and Lorenz T Biegler. On the implementation of an interior-point filter line-search algorithm for large-scale nonlinear programming. *Mathematical programming*, 106:25–57, 2006. 2, 7

[58] Patrick M. Wensing, Michael Posa, Yue Hu, Adrien Escande, Nicolas Mansard, and Andrea Del Prete. Optimization-based control for dynamic legged robots. *IEEE Transactions on Robotics*, 40:43–63, 2024. 11

[59] Wikipedia contributors. Semi-implicit euler method — Wikipedia, the free encyclopedia, 2023. URL https://en.wikipedia.org/wiki/Semi-implicit_Euler_method. [Online; accessed 24-May-2023]. 18

[60] William Yang and Michael Posa. Dynamic on-palm manipulation via controlled sliding. *arXiv preprint arXiv:2405.08731*, 2024. 9, 12

[61] Jane J. Ye and Jinchuan Zhou. First-order optimality conditions for mathematical programs with second-order cone complementarity constraints. *SIAM Journal on Optimization*, 26(4):2820–2846, 2016. 2

[62] K. Yunt and C. Glocker. Trajectory optimization of mechanical hybrid systems using sumt. In *9th IEEE International Workshop on Advanced Motion Control, 2006.*, pages 665–671, 2006. 1

APPENDIX A
PROOF OF THEOREM 8

Proof: As x_k converges to $x^\star$, we have $\|p_k\| = \|x_{k+1} - x_k\| \to 0$. By definition, there exists a constant K, such that $\|p_k\| < \Delta_{\min}$ for all $k > K$, indicating that the trust-region constraint becomes inactive in the subproblem. In this proof, we consider only equality constraints, and the proof technique for inequality constraints is similar and will be addressed at the end. In this case, the merit function (8) becomes:

$$\phi_1(x;\mu) \triangleq J(x) + \sum_{i\in\mathcal{E}} \mu_i |c_i(x)|, \tag{18}$$

where $x \in \mathbb{R}^n$, $\mathcal{E}$ denotes the equality constraints set and $|\mathcal{E}| = m_\mathcal{E}$. First, we rewrite the trust-region subproblem (9) to directly optimize over x_{k+1} instead of p_k:

$$x_{k+1} = \arg\min_x \{q_{\mu,k}(x) = J(x) + \sum_{i\in\mathcal{E}} \mu_i |c_i(x_k) + \nabla c_i(x_k)^\top (x - x_k)|\}. \tag{19}$$

For the ith constraint $c_i \in \mathcal{E}$, we define $w_{k,i} \in \mathbb{R}$ as:

$$w_{k,i} \triangleq c_i(x_k) + \nabla c_i(x_k)^\top (x_{k+1} - x_k), \tag{20}$$

Since the l_1 norm $|\cdot|$ is piecewise smooth, we separate the scalar $w_{k,i}$ into three smooth sets:

$$\{w_{k,i} < 0\}, \{w_{k,i} = 0\}, \{w_{k,i} > 0\}. \tag{21}$$

Extending to the vector of all equality constraints $w_k = \begin{bmatrix} w_{k,1} & \dots & w_{k,m_\mathcal{E}} \end{bmatrix} \in \mathbb{R}^{m_\mathcal{E}}$, we can divide $\{w_k\}_{k=K}^\infty$ to $3^{m_\mathcal{E}}$ smooth sets without overlapping.

Since the number of such sets is finite, there must exist a subsequence $\{w_{n_k}\}_{k=1}^\infty$ that belongs to one of these sets, denoting as S. Without loss of generality, we assume that in this set, the first m_1 components of w_{n_k} are positive, the $m_1 + 1$ to $m_1 + m_2$ components are negative, and the remaining components are zero.

For any $p \in \mathbb{R}^n$, we define the directional derivatives[1]:

$$D(\phi_1(x^\star;\mu);p) \triangleq \lim_{t\downarrow 0} \frac{\phi_1(x^\star + tp;\mu) - \phi_1(x^\star;\mu)}{t}, \tag{22}$$

$$D(q_{\mu,n_k}(x_{n_k+1});p) \triangleq \lim_{t\downarrow 0} \frac{q_{\mu,n_k}(x_{n_k+1} + tp) - q_{\mu,n_k}(x_{n_k+1})}{t}. \tag{23}$$

Since x_{n_k+1} is the global minimizer of the convex function $q_{\mu,n_k}(x)$, we have $D(q_{\mu,n_k}(x_{n_k+1});p) \geq 0$. On the other hand,

$$D(q_{\mu,n_k}(x_{n_k+1});p) = \lim_{t\downarrow 0} \frac{J(x_{n_k+1} + tp) + \sum_{i\in\mathcal{E}} \mu_i |w_{n_k,i} + t\nabla c_i(x_{n_k})^\top p| - J(x_{n_k+1}) - \sum_{i\in\mathcal{E}} \mu_i |w_{n_k,i}|}{t} \tag{24}$$

$$= \nabla J(x_{n_k+1})^\top p + \lim_{t\downarrow 0} \frac{\sum_{i\in\mathcal{E}} \mu_i |w_{n_k,i} + t\nabla c_i(x_{n_k})^\top p| - \sum_{i\in\mathcal{E}} \mu_i |w_{n_k,i}|}{t}. \tag{25}$$

$$= \nabla J(x_{n_k+1})^\top p + \sum_{i=1}^{m_1} \mu_i \nabla c_i(x_{n_k})^\top p - \sum_{i=m_1+1}^{m_1+m_2} \mu_i \nabla c_i(x_{n_k})^\top p + \sum_{i=m_1+m_2+1}^{m} \mu_i |\nabla c_i(x_{n_k})^\top p|. \tag{26}$$

$$\tag{27}$$

Using the fact that:

$$\lim_{t\downarrow 0} \frac{|a + tb| - |a|}{t} = \begin{cases} b, & a > 0 \\ -b, & a < 0 \\ |b|, & a = 0 \end{cases} \tag{28}$$

Since $J(x)$ is continuous differentiable and ∇c_i is Lipschitz, as $k \to \infty$:

$$\nabla J(x_{n_k+1}) \to \nabla J(x^\star), \tag{29}$$

$$\nabla c_i(x_{n_k}) \to \nabla c_i(x^\star). \tag{30}$$

Thus,

$$D(q_{\mu,n_k}(x_{n_k+1});p) \xrightarrow[k\to\infty]{} \nabla J(x^\star)^\top p + \sum_{i=1}^{m_1} \mu_i \nabla c_i(x^\star)^\top p - \sum_{i=m_1+1}^{m_1+m_2} \mu_i \nabla c_i(x^\star)^\top p + \sum_{i=m_1+m_2+1}^{m} \mu_i |(\nabla c_i(x^\star)^\top p)| \geq 0. \tag{31}$$

[1]We use $\downarrow$ for simplicity, but this is equivalent to $\to$ because one can easily choose p as $-p$ if $t < 0$.

The last inequality comes from the fact that if all the elements in a scaler sequence are nonnegative, then the limitation of the sequence is nonnegative.

Now we turn to calculate $D(\phi_1(x^\star;\mu);p)$. One should be careful: since S is not closed, even the first m_1 components of w_{n_k} are always positive, the corresponding components in the converged point $w^\star = \begin{bmatrix} c_1(x^\star) & \dots & c_{m_\mathcal{E}}(x^\star) \end{bmatrix}$ will possibly converge to 0. Similar things happen to the negative part. However, the $m_1+m_2+1 \sim m_\mathcal{E}$ components of $w^\star$ will always be 0. To fix this, we assume that, for the $1 \sim m_1$ components of $w^\star$, $1 \sim m_{1,+}$ are positive, while $m_{1,+}+1 \sim m_1$'s components of $w^\star$ become zero. Similarly, $m_1+1 \sim m_1+m_{2,-}$ are negative, while $m_1+m_{2,-}+1 \sim m_1+m_2$'s components of $w^\star$ become zero. Thus,

$$D(\phi_1(x^\star;\mu);p) = \lim_{t\downarrow 0} \frac{J(x^\star+tp)+\sum_{i\in\mathcal{E}}\mu_i|c_i(x^\star+tp)|-J(x^\star)-\sum_{i\in\mathcal{E}}\mu_i|c_i(x^\star)|}{t} \tag{32}$$

$$= \nabla J(x^\star)^\top p + \sum_{i=1}^{m_{1,+}} \mu_i \nabla c_i(x^\star)^\top p + \sum_{i=m_{1,+}+1}^{m_1} \mu_i |\nabla c_i(x^\star)^\top p| \tag{33}$$

$$- \sum_{i=m_1+1}^{m_1+m_{2,-}} \mu_i \nabla c_i(x^\star)^\top p + \sum_{i=m_1+m_{2,-}+1}^{m_1+m_2} \mu_i |\nabla c_i(x^\star)^\top p| + \sum_{i=m_1+m_2+1}^{m_\mathcal{E}} \mu_i |\nabla c_i(x^\star)^\top p|. \tag{34}$$

$$\geq \nabla J(x^\star)^\top p + \sum_{i=1}^{m_1} \mu_i \nabla c_i(x^\star)^\top p - \sum_{i=m_1+1}^{m_1+m_2} \mu_i \nabla c_i(x^\star)^\top p + \sum_{i=m_1+m_2+1}^{m_\mathcal{E}} \mu_i |\nabla c_i(x^\star)^\top p|. \tag{35}$$

$$\geq 0 \tag{36}$$

For inequality constraints we define $w_{k,i}$ as the same. We still separate $\mathbb{R}$ into

$$\{w_{k,i}<0\}, \{w_{k,i}=0\}, \{w_{k,i}>0\} \tag{37}$$

and similarly, we have

$$\lim_{t\downarrow 0} \frac{|a+tb|^- - |a|^-}{t} = \begin{cases} 0, & a>0 \\ -b, & a<0 \\ |b|^-, & a=0 \end{cases} \tag{38}$$

Other parts of the proof are the same. Consequently $D(\phi_1(x^\star;\mu);p) \geq 0 \quad \forall p$, we conclude the proof. ■

APPENDIX B
CONTACT-IMPLICIT FORMULATIONS

In this section, we provide detailed contact-implicit optimization problem formulations for the five tasks, including tracking objectives, dynamics, contact constraints, and other constraints. We believe this presentation is not only valuable for understanding the principles of contact-implicit modeling, but also represents a necessary step for the model-based community. Interested readers are encouraged to refer to these formulations to test these same problems using their own algorithms. All problem formulations are open-sourced alongside CRISP.

A. Cartpole with Softwalls

Dynamics constraints. Denote the full states of the system as:

$$v = [\underbrace{x, \theta, \dot{x}, \dot{\theta}}_{\text{states}}, \underbrace{u, \lambda_1, \lambda_2}_{\text{control}}],$$

and we use subscript i to denote the corresponding variable at time stamp i. The dynamics can be written as:

$$(m_c+m_p)\ddot{x} + m_p \ell \ddot{\theta}\cos\theta - (u-\lambda_1+\lambda_2) = 0, \tag{39}$$

$$m_p \ell \ddot{\theta} - (\lambda_2-\lambda_1-m_p\ddot{x})\cos\theta - m_p g \sin\theta = 0. \tag{40}$$

Here, x denotes the cart position, while m_c and m_p represent the mass of the cart and the pole end, respectively. It is important to note that the contact forces λ_1 and λ_2 are incorporated into the dynamics equations despite their discrete nature. These forces are automatically triggered and controlled by the complementarity constraints introduced below, which exemplifies the

principle and power of contact-implicit formulation. We utilize the Semi-Implicit Euler method [59] to discretize the dynamics with dt. At time stamp i:

$$x_{i+1} - x_i - \dot{x}_{i+1} dt = 0, \tag{41}$$
$$\theta_{i+1} - \theta_i - \dot{\theta}_{i+1} dt = 0, \tag{42}$$
$$\dot{x}_{i+1} - x_i - \ddot{x}_i dt = 0, \tag{43}$$
$$\dot{\theta}_{i+1} - \theta_i - \ddot{\theta}_i dt = 0. \tag{44}$$

In the above dynamics constraints, $\ddot{x}$ and $\ddot{\theta}$ can be simply derived from (39)-(40).

Contact constraints. The complementary constraints from the contact are:

$$0 \leq \lambda_1 \perp \left(\frac{\lambda_1}{k_1} + d_1 - x - \ell \sin\theta \right) \geq 0, \tag{45}$$
$$0 \leq \lambda_2 \perp \left(\frac{\lambda_2}{k_2} + d_2 + x + \ell \sin\theta \right) \geq 0. \tag{46}$$

This complementarity constraint is constructed using the relationship between the pole's position and the wall, along with the contact force. Its underlying meaning is that the wall can only provide unidirectional force. When there is no contact, λ is zero. Upon contact, the force becomes proportional to the compression distance with coefficients k.

Initial Constraints. The optimization problem is also subject to equality constraints that specify the initial state.

$$x_0 = x_{initial} \tag{47}$$
$$\theta_0 = \theta_{initial} \tag{48}$$
$$\dot{x}_0 = \dot{x}_{initial} \tag{49}$$
$$\dot{\theta}_0 = \dot{\theta}_{initial} \tag{50}$$

Cost. For the objective function, we opt to minimize both the control effort and the terminal tracking error, each expressed in quadratic form. It is worth noting that intermediate tracking loss is not incorporated into our formulation. This decision stems from the general absence of trivial reference trajectories in contact-involved trajectory optimization tasks. Our goal is to generate a feasible trajectory from scratch that drives the robot as close as possible to the desired terminal state.

$$f = \frac{1}{2} \sum_{i=1}^{N-1} v_i^T \begin{bmatrix} 0 & 0 \\ 0 & R \end{bmatrix} v_i + \frac{1}{2} v_N^T \begin{bmatrix} Q & 0 \\ 0 & 0 \end{bmatrix} v_N, \tag{51}$$

where $Q \in \mathbb{R}^{4\times4}$ and $R \in \mathbb{R}^{3\times3}$ are weighting matrices for the state and control variables. All the default values of the parameters are provided in the example problems of CRISP.

B. Push Box

We use p_x, p_y, and θ to represent the position and orientation of the box in world frame $\{w\}$, while the contact position (c_x, c_y) and contact force λ at each facet is defined in the body frame $\{b\}$. We assume that at any given moment, there is only one point of contact between the pusher and the box with only the corresponding normal force applied, and the entire process is quasi-static. The positive direction of the applied forces is defined with respect to the body frame $\{b\}$ of the robot.

Dynamics constraints. In this work, following [26], we adapt the commonly used ellipsoidal approximation of the limit surface to model the interaction between the contact force applied by the pusher and the resulting spatial slider velocity. The model captures the principle of the motion of the box while keeping its simplicity. Denote the full states of the system as:

$$v = \underbrace{[p_x, p_y, \theta}_{\text{states}}, \underbrace{c_x, c_y, \lambda_{1,y}, \lambda_{2,x}, \lambda_{3,y}, \lambda_{4,x}]}_{\text{control}}.$$

Then, the push box dynamics can be writen as:

$$\dot{p}_x = \frac{1}{\mu m g} . \left[(\lambda_{2,x} + \lambda_{4,x}) \cos\theta - (\lambda_{1,y} + \lambda_{3,y}) \sin\theta \right], \tag{52}$$
$$\dot{p}_y = \frac{1}{\mu m g} . \left[(\lambda_{2,x} + \lambda_{4,x}) \sin\theta + (\lambda_{1,y} + \lambda_{3,y}) \cos\theta \right], \tag{53}$$
$$\dot{\theta} = \frac{1}{cr\mu m g} . \left[-c_y(\lambda_{2,x} + \lambda_{4,x}) + c_x(\lambda_{1,y} + \lambda_{3,y}) \right], \tag{54}$$

where $c \in [0, 1]$ is the integration constant of the box, and r is the characteristic distance, typically chosen as the max distance between a contact point and origin of frame $\{b\}$. Discretize the continuous dynamics with the explicit Euler method, we get:

$$p_{x,k+1} = p_{x,k} + \dot{p}_{x,k} dt, \tag{55}$$
$$p_{y,k+1} = p_{y,k} + \dot{p}_{y,k} dt, \tag{56}$$
$$\theta_{k+1} = \theta_k + \dot{\theta}_k dt. \tag{57}$$

Contact constraints. First, we ensure that the contact force can only be applied through the contact point (c_x, c_y), and only pointed inward the box, utilizing the contact-implicit formulation:

$$0 \leq \lambda_{1,y} \perp (c_y + b) \geq 0, \tag{58}$$
$$0 \leq \lambda_{2,x} \perp (c_x + a) \geq 0, \tag{59}$$
$$0 \leq -\lambda_{3,y} \perp (b - c_y) \geq 0, \tag{60}$$
$$0 \leq -\lambda_{4,x} \perp (a - c_x) \geq 0. \tag{61}$$

Moreover, to prevent the simultaneous application of forces on adjacent edges at corners, we introduce additional complementarity constraints. These constraints ensure that at any given time, only one force can be active. This is formulated as follows:

$$0 \leq \lambda_{1,y} \perp \lambda_{2,x} \geq 0, \tag{62}$$
$$0 \leq \lambda_{1,y} \perp -\lambda_{3,y} \geq 0, \tag{63}$$
$$0 \leq \lambda_{1,y} \perp -\lambda_{4,x} \geq 0, \tag{64}$$
$$0 \leq \lambda_{2,x} \perp -\lambda_{3,y} \geq 0, \tag{65}$$
$$0 \leq \lambda_{2,x} \perp -\lambda_{4,x} \geq 0, \tag{66}$$
$$0 \leq -\lambda_{3,y} \perp -\lambda_{4,x} \geq 0. \tag{67}$$

Initial Constraints. We add equality constraints to enforce the initial state of the box with $(\bar{p}_{x,0}, \bar{p}_{y,0}, \bar{\theta}_0)$. Note that the initial condition is set to zero in this task.

Cost. Similar to the setting of cartpole with walls, we adopt a quadratic objective function to penalize the total four contact forces λ and the distance to the desired terminal condition $\bar{v}_N$.

$$f = \frac{1}{2} \sum_{i=1}^{N-1} v_i^T \begin{bmatrix} 0 & 0 \\ 0 & R \end{bmatrix} v_i + \frac{1}{2} (v_N - \bar{v}_N)^T \begin{bmatrix} Q & 0 \\ 0 & 0 \end{bmatrix} (v_N - \bar{v}_N), \tag{68}$$

C. Transport

Dynamics constraints: Denote all states as

$$v = [\underbrace{x_1, x_2, \dot{x}_1, \dot{x}_2, p, q}_{\text{states}} \underbrace{f, u}_{\text{control}}].$$

f is the friction between the two blocks with friction coefficients μ_1; p and q are two slack variables for modeling the direction of the f, which is decided by the relative movements between the two blocks. The positive directions of f and x are both horizontally to the right. The dynamics for this problem are straightforward:

$$m_1 \ddot{x}_1 = f, \tag{69}$$
$$m_2 \ddot{x}_2 = u - f. \tag{70}$$

To avoid cargo m_1 from falling off the truck m_2, we need:

$$-l_0 \leq x_1 - x_2 \leq l_0. \tag{71}$$

Contact constraints. The control of the magnitude and direction of friction requires careful handling. f is not an active force; it needs to be indirectly controlled through the manipulation of u. This indirect control determines the friction's direction, magnitude, and whether it manifests as static or kinetic friction.

$$\dot{x}_2 - \dot{x}_1 = p - q, \tag{72}$$
$$0 \leq p \perp q \geq 0, \tag{73}$$
$$0 \leq p \perp (\mu_1 m_1 g - f) \geq 0, \tag{74}$$
$$0 \leq q \perp (f + \mu_1 m_1 g) \geq 0. \tag{75}$$

This establishes the relationship between relative velocity and friction force while encoding the transition between static and kinetic friction in a clean way.

Initial constraints. We add equality constraints to enforce the initial pose and velocity of the cart and the payload.

Cost. We adopt a quadratic objective function to penalize the active force u and the terminal pos and velocity tracking error.

D. Push T

Contact-Implicit Formulation: Denote the contact point in body frame (located at the COM of T block) as (c_x, c_y), we have 8 different contact mode that exhibit their specific complementarity constraints.

$$-2l \leq c_x \leq 2l, -d_c l \leq c_y \leq (4-d_c)l, \tag{76}$$
$$\lambda_1 \neq 0 \Rightarrow c_y - (4-d_c)l = 0, \tag{77}$$
$$\lambda_2 \neq 0 \Rightarrow |c_x - 2l| + |c_y - (4-d_c)l| + |c_y - (3-d_c)l| - l = 0, \tag{78}$$
$$\lambda_3 \neq 0 \Rightarrow |c_x - 2l| + |c_x - 0.5l| + |c_y - (3-d_c)l| - 1.5l = 0, \tag{79}$$
$$\lambda_4 \neq 0 \Rightarrow |c_x - 0.5l| + |c_y - (3-d_c)l| + |c_y + d_c l| - 3l = 0, \tag{80}$$
$$\lambda_5 \neq 0 \Rightarrow |c_y + d_c l| + |c_x + 0.5l| + |c_x - 0.5l| - l = 0, \tag{81}$$
$$\lambda_6 \neq 0 \Rightarrow |c_x + 0.5l| + |c_y - (3-d_c)l| + |c_y + d_c l| - 3l = 0, \tag{82}$$
$$\lambda_7 \neq 0 \Rightarrow |c_x + 2l| + |c_x + 0.5l| + |c_y - (3-d_c)l| - 1.5l = 0, \tag{83}$$
$$\lambda_8 \neq 0 \Rightarrow |c_x + 2l| + |c_y - (4-d_c)l| + |c_y - (3-d_c)l| - l = 0. \tag{84}$$

Designate the positive direction of contact forces as rightward for the x axis and upward for the y axis. And introduce the following slack variables v and w complementing each other to smooth the absolute operation $|\cdot|$ as we did in the transport example:

$$v_1 - w_1 = c_x - 2l, \tag{85}$$
$$v_2 - w_2 = c_y - (4-d_c)l, \tag{86}$$
$$v_3 - w_3 = c_y - (3-d_c)l, \tag{87}$$
$$v_4 - w_4 = c_x - 0.5l, \tag{88}$$
$$v_5 - w_5 = c_y + d_c l, \tag{89}$$
$$v_6 - w_6 = c_x + 0.5l, \tag{90}$$
$$v_7 - w_7 = c_x + 2l, \tag{91}$$
$$0 \leq v_1 \perp w_1 \geq 0, \tag{92}$$
$$0 \leq v_2 \perp w_2 \geq 0, \tag{93}$$
$$0 \leq v_3 \perp w_3 \geq 0, \tag{94}$$
$$0 \leq v_4 \perp w_4 \geq 0, \tag{95}$$
$$0 \leq v_5 \perp w_5 \geq 0, \tag{96}$$
$$0 \leq v_6 \perp w_6 \geq 0, \tag{97}$$
$$0 \leq v_7 \perp w_7 \geq 0, \tag{98}$$
$$0 \leq v_8 \perp w_8 \geq 0, \tag{99}$$

which can be equivalently written as:

$$v_1 + w_1 = |c_x - 2l|, \tag{100}$$
$$v_2 + w_2 = |c_y - (4-d_c)l|, \tag{101}$$
$$v_3 + w_3 = |c_y - (3-d_c)l|, \tag{102}$$
$$v_4 + w_4 = |c_x - 0.5l|, \tag{103}$$
$$v_5 + w_5 = |c_y + d_c l|, \tag{104}$$
$$v_6 + w_6 = |c_x + 0.5l|, \tag{105}$$
$$v_7 + w_7 = |c_x + 2l|, \tag{106}$$
$$\tag{107}$$

Then we have the complementarity constraints:

$$0 \leq -\lambda_1 \perp (4 - d_c)l - c_y \geq 0, \tag{108}$$
$$0 \leq -\lambda_2 \perp v_1 + w_1 + v_2 + w_2 + v_3 + w_3 - l \geq 0, \tag{109}$$
$$0 \leq \lambda_3 \perp v_1 + w_1 + v_3 + w_3 + v_4 + w_4 - 1.5l \geq 0, \tag{110}$$
$$0 \leq -\lambda_4 \perp v_3 + w_3 + v_4 + w_4 + v_5 + w_5 - 3l \geq 0, \tag{111}$$
$$0 \leq \lambda_5 \perp v_4 + w_4 + v_5 + w_5 + v_6 + w_6 - l \geq 0, \tag{112}$$
$$0 \leq \lambda_6 \perp v_3 + w_3 + v_5 + w_5 + v_6 + w_6 - 3l \geq 0, \tag{113}$$
$$0 \leq \lambda_7 \perp v_3 + w_3 + v_6 + w_6 + v_7 + w_7 - 1.5l \geq 0, \tag{114}$$
$$0 \leq \lambda_8 \perp v_2 + w_2 + v_3 + w_3 + v_7 + w_7 - l \geq 0. \tag{115}$$

Furthermore, if we want to avoid simultaneous application of forces on adjacent edges at corners, we introduce additional complementarity constraints. These constraints ensure that at any given time, only one force can be active:

$$\lambda_i \perp \lambda_j, \forall i, j = 1 \dots 8, \text{and } i \neq j. \tag{116}$$

With the complementarity constraints derived above, we can write the unified dynamics regardless of the contact point.

$$\dot{p}_x = \frac{1}{\mu m g}.\left[(\lambda_2 + \lambda_4 + \lambda_6 + \lambda_8)\cos\theta - (\lambda_1 + \lambda_3 + \lambda_5 + \lambda_7)\sin\theta\right], \tag{117}$$
$$\dot{p}_y = \frac{1}{\mu m g}.\left[(\lambda_2 + \lambda_4 + \lambda_6 + \lambda_8)\sin\theta + (\lambda_1 + \lambda_3 + \lambda_5 + \lambda_7)\cos\theta\right], \tag{118}$$
$$\dot{\theta} = \frac{1}{cr\mu m g}.\left[-c_y(\lambda_2 + \lambda_4 + \lambda_6 + \lambda_8) + c_x(\lambda_1 + \lambda_3 + \lambda_5 + \lambda_7)\right], \tag{119}$$

where $c \in [0, 1]$ is the integration constant of the box, and r is the characteristic distance, typically chosen as the max contact distance.

E. Hopper

The hopper exhibits two-phase dynamics, which we will unify through the complementarity constraints to decide whether it is in the fly or stance.

Dynamics constraints. We define the full states of the hopping robot as:

$$v = [\underbrace{p_x, p_y, q_x, q_y, \theta, r, \dot{p}_x, \dot{p}_y}_{\text{states}}, \underbrace{u_1, u_2}_{\text{control}}].$$

(p_x, p_y) and (q_x, q_y) are the positions of the head and tail of the hopper respectively. r is the leg compression distance. u_1 and u_2 are the controls for leg angular velocity and thrust, each active only in one phase. Then, the hopper's dynamics can be written as:

- Flight Phase:

$$\ddot{p}_x = 0, \tag{120a}$$
$$\ddot{p}_y = -g, \tag{120b}$$
$$q_x = p_x + l_0 \sin\theta, \tag{120c}$$
$$q_y = p_y - l_0 \cos\theta, \tag{120d}$$
$$\dot{\theta} = u_1. \tag{120e}$$

- Stance Phase:
  In the stance phase, (q_x, q_y) is fixed at the contact point, and we can get the following dynamics equations:

$$m\ddot{p}_x + u_2 \sin\theta = 0, \tag{121a}$$
$$m\ddot{p}_y - u_2 \cos\theta + mg = 0, \tag{121b}$$
$$(l_0 - r)\cos\theta - p_y + q_y = 0, \tag{121c}$$
$$(l_0 - r)\sin\theta - q_x + p_x = 0. \tag{121d}$$
$$(l_0 - r)^2 - (p_x - q_x)^2 - (p_y - q_y)^2 = 0, \tag{121e}$$

Also, q_x and q_y are fixed by:

$$\dot{q}_x = 0, \tag{122a}$$
$$\dot{q}_y = 0. \tag{122b}$$

Since during the flight phase, r remains 0, while u_2 can only be greater than 0 during the stance phase. We can observe that (120a)-(120d) are actually the same as (121a)-(121d), except that they involve these phase-specific variables. To unify the dynamics into a single optimization problem, we construct the following complementarity constraints:

$$0 \leq r \perp q_y \geq 0, \tag{123a}$$
$$0 \leq u_2 \perp q_y \geq 0, \tag{123b}$$
$$0 \leq u_1^2 \perp r \geq 0. \tag{123c}$$

These constraints ensure that the leg maintains its original length during the flight phase while allowing contraction ($r \geq 0$) when in contact with the ground. These conditions are complementary, enabling us to control the switching of dynamics between phases. Specifically, we transform the dynamics of different phases to:

$$m\ddot{p}_x + u_2 \sin\theta = 0, \tag{124}$$
$$m\ddot{p}_y - u_2 \cos\theta + mg = 0, \tag{125}$$
$$(l_0 - r)\cos\theta - p_y + q_y = 0, \tag{126}$$
$$(l_0 - r)\sin\theta - q_x + p_x = 0, \tag{127}$$
$$r.\dot{q}_x = 0, \tag{128}$$
$$r.\dot{q}_y = 0, \tag{129}$$
$$q_y.(\theta - u_1) = 0, \tag{130}$$
$$r.\left((l_0 - r)^2 - (p_x - q_x)^2 - (p_y - q_y)^2\right) = 0. \tag{131}$$

These unified dynamics, together with the complementary constraints (123a)-(123c), provide a way to write the hopper problem into one optimization problem in the contact-implicit format. Discretize the dynamics, we get:

$$m\frac{\dot{p}_{x,k+1} - \dot{p}_{x,k}}{dt} + u_{2,k}\sin\theta_k = 0, \tag{132}$$
$$m\frac{\dot{p}_{y,k+1} - \dot{p}_{y,k}}{dt} - u_{2,k}\cos\theta_k + mg = 0, \tag{133}$$
$$(l_0 - r_k)\cos\theta_k - p_{y,k} + q_{y,k} = 0, \tag{134}$$
$$(l_0 - r_k)\sin\theta_k - q_{x,k} + p_{x,k} = 0, \tag{135}$$
$$r_k.\frac{q_{x,k+1} - q_{x,k}}{dt} = 0, \tag{136}$$
$$r_k.\frac{q_{y,k+1} - q_{y,k}}{dt} = 0, \tag{137}$$
$$r_k.\left((l_0 - r_k)^2 - (p_{x,k} - q_{x,k})^2 - (p_{y,k} - q_{y,k})^2\right) = 0. \tag{138}$$

Contact constraints. We introduce additional constraints to ensure physical consistency and feasibility:

$$0 \leq r \leq r_0. \tag{139}$$

The leg contraction r within a feasible range. The constant r_0 represents the maximum allowable contraction and is chosen such that $r_0 < l_0$, where l_0 is the original leg length.

These additional constraints, in conjunction with the complementarity constraint, provide a comprehensive formulation that captures the essential physical characteristics of the hopping robot while maintaining the unified representation of both flight and stance phases.

Initial constraints. The hopper is released from 1.5 m height with zero initial speed and the leg vertical.

Cost. The cost is a quadratic loss to force the robot to jump to and stop at 2 m with zero height while penalizing the control effort of the angular velocity u_1 and thrust u_2.

F. Waiter

Dynamics constraints. Denote the full states in the waiter problem:

$$v = [\underbrace{x_1, x_2, \dot{x}_1, \dot{x}_2, v, w, p, q}_{\text{states}}, \underbrace{\lambda_N, u, f_p, f_t, N}_{\text{control}}].$$

In this scenario, (x_1, x_2) and $(\dot{x}_1, \dot{x}_2)$ represent the position and velocity of the plate and pusher, respectively. The variables $z, w, p,$ and q are slack variables used to model the friction between the plate and table, as well as between the pusher and plate, similar to the transport example. For control, we have direct control over the normal force applied by the pusher, denoted as λ_N, and the horizontal thrust, u. Additionally, there is indirect control over the frictional forces between the plate and table, f_t, and between the pusher and plate, f_p. The support force exerted by the table is N. The system dynamics are described by the following equations:

$$m_2\ddot{x}_2 = u - f_p, \tag{140a}$$
$$m_1\ddot{x}_1 = f_p - f_t, \tag{140b}$$
$$N + \lambda_N = m_1 g, \tag{140c}$$
$$\lambda_N(x_2 - x_1 + l_0) \leq m_1 g l_0, \tag{140d}$$
$$N \geq 0,\ \lambda_N \geq 0, \tag{140e}$$
$$x_2 \geq 0,\ x_2 - x_1 \leq l_0. \tag{140f}$$

Here, (140c) ensures vertical force balance. (140d) prevents tilting around the leftmost contact point between the table and plate. Finally, (140f) restricts the pusher's position to remain clear of the table while staying on the overhanging part of the plate.

Contact constraints. The following complementarity constraints control the magnitude and direction of the friction f_t and f_p

$$\dot{x}_1 = z - w, \tag{141}$$
$$\dot{x}_2 - \dot{x}_1 = p - q, \tag{142}$$
$$0 \leq z \perp w \geq 0, \tag{143}$$
$$0 \leq p \perp q \geq 0, \tag{144}$$
$$0 \leq z \perp (N\mu_1 - f_t) \geq 0, \tag{145}$$
$$0 \leq w \perp (N\mu_1 + f_t) \geq 0, \tag{146}$$
$$0 \leq p \perp (\mu_2\lambda_N - f_p) \geq 0, \tag{147}$$
$$0 \leq q \perp (\mu_2\lambda_N + f_p) \geq 0. \tag{148}$$

These equations ensure the friction cone constraints, which automatically manage the magnitude, nature (static or kinetic), and direction of the frictions.

Initial constraints. In the waiter problem, the initial state is defined with the pusher positioned next to the table, and the 14-meter-long plate has a 1-meter overhang.

Cost. The objective is to achieve a terminal position and velocity of the plate and pusher, such that the plate is pulled out until its COM aligns with the pusher next to the table, while also minimizing control efforts.

APPENDIX C
VISUALIZATION OF NUMERICAL RESULTS

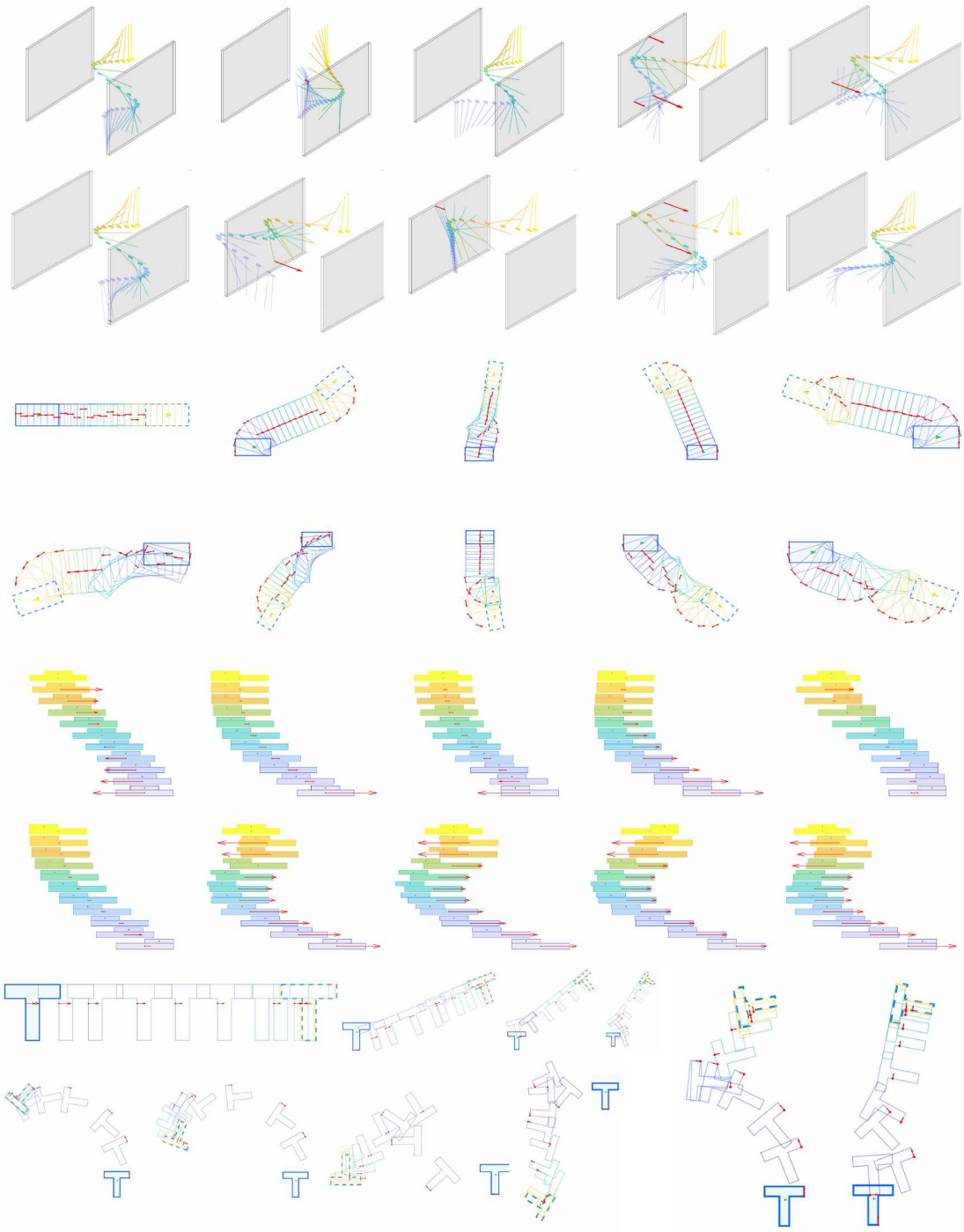


Fig. 9: Visualization of more trajectories solved by CRISP in cartpole with soft walls (row 1-2), push box (row 3-4), payload transport (row 5-6), push T (row 7-8) under different initial conditions.