2026 IEEE International Conference on Robotics and Automation (ICRA 2026)
June 1-5, 2026. Vienna, Austria

The Trajectory Bundle Method: Unifying Sequential-Convex Programming and Sampling-Based Trajectory Optimization

Kevin Tracy[1], John Z. Zhang[1], Jon Arrizabalaga[1],
Stefan Schaal[2], Yuval Tassa[3], Tom Erez[3], and Zachary Manchester[1]

***Abstract*— We present a unified framework for solving trajectory optimization problems in a derivative-free manner through the use of sequential convex programming. Traditionally, nonconvex optimization problems are solved by forming and solving a sequence of convex optimization problems, where the cost and constraint functions are approximated locally through Taylor series expansions. This presents a challenge for functions where differentiation is expensive or unavailable. In this work, we present a derivative-free approach to form these convex approximations by computing samples of the dynamics, cost, and constraint functions and letting the solver interpolate between them. Our framework includes sample-based trajectory optimization techniques like model-predictive path integral (MPPI) control as a special case and generalizes them to enable features like multiple shooting and general equality and inequality constraints that are traditionally associated with derivative-based sequential convex programming methods. The resulting framework is simple, flexible, and capable of solving a wide variety of practical motion planning and control problems.**

I. Introduction

Linear dynamical systems of the form $x_+ = Ax + Bu$ underpin many of the foundational methods in modern optimal control. Ideas such as the Linear-Quadratic Regulator (LQR) and convex trajectory optimization can reason about dynamical systems of this form in a way that is globally optimal [1], [2], [3]. As a result, these techniques are often applied to nonlinear systems where the dynamics are locally approximated as linear around a linearization point [4]. In many cases, this approximation is appropriate given the function is not being evaluated too far from where the approximation was formed. When used appropriately, this method of linearizing nonlinear systems can be extremely effective in practice, even for highly nonlinear systems. The two caveats here are that the nonlinear system must be both smooth and differentiable.

For many systems, such as robotic arms, quadrotors, and wheeled vehicles, this assumption of smooth differentiability is reasonable. For rigid-body dynamics, there are specialized methods for computing derivatives of the continuous-time dynamics in a fast and efficient way [5]. However, for more complex dynamics models, there are scenarios where these derivatives are unavailable, prohibitively expensive to compute, or unreliable. If the dynamics model is learned from data, the approximation of the dynamics function may be good, while the approximation of the derivatives may be very poor. This scenario is often explored in the context of model-predictive path-integral (MPPI) control, where a learned simulator is only used to produce parallelized simulations [6], [7].

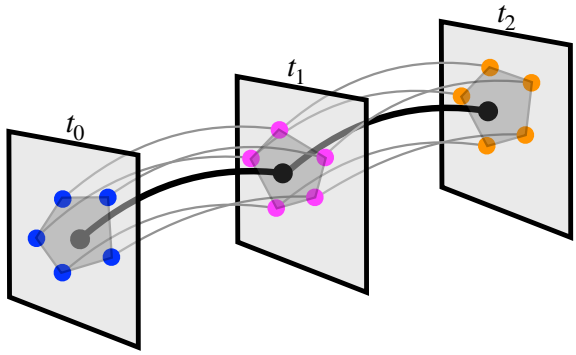

Fig. 1: The Trajectory Bundle Method (TBM) is a derivative-free framework capable of solving both single-shooting sampling-based MPC problems *and* general multiple-shooting trajectory optimization problems with constraints. TBM finds the optimal state and control sequences (bolded black trajectory) by computing samples (colored points) at each knot point (ts) around the current solution and using convex optimization to linearly interpolate between the samples (dark gray regions).

Another scenario in which derivatives are unavailable or unusable is in the presence of systems that make or break contact. While there has been a lot of recent interest in making contact simulation differentiable [8], [9], [10], [11], there remains a strong need for optimal control methods that do not rely on these derivatives at all.

A recent trend in robotic simulation is the introduction of simulators that can be run on accelerators for massively parallel simulation. Popular simulators like Isaac Sim [12], [13], Brax [8], and MuJoCo XLA (MJX) [14], are all capable of running thousands of simulations in parallel. This paper leverages the innovations in parallel simulation to motivate a new derivative-free optimal control paradigm where simulation rollouts are used to fully describe the dynamics and cost landscapes present in the problem. We introduce the trajectory bundle method for solving nonconvex trajectory optimization problems, which uses interpolated trajectories instead of derivative-based linearizations to approximate the cost, dynamics, and constraint functions in the problem. The result is a simple and robust trajectory optimization framework that can fully utilize parallelized simulation without requiring any derivatives.

Our specific contributions in this paper are the following:

- A unified framework, which we refer to as the trajectory bundle method, for solving general trajectory optimization problems using derivative-free sequential-convex

[1]Robotics Institute (RI), Carnegie Mellon University (CMU)
[2]Google Intrinsic
[3]Google Deepmind

programming.

- A method for approximating general nonlinear or non-convex cost and constraint functions through sampling and linear interpolation.
- A set of numerical experiments demonstrating the effectiveness of the trajectory bundle method and its connections to SCP and MPPI.

The remainder of the paper is organized as follows: we first review related literature on derivative-free optimization, sequential-convex programming, and MPPI in Section II. Next, we introduce relevant background on affine function approximation and its application to constrained optimization in Section III. In Section IV, we describe the trajectory bundle method in a general multiple-shooting framework and a single-shooting special case that is equivalent to MPPI. Finally, we present an array of numerical experiments in Section V and point avenues of future research in VI.

II. Related Work

In order to contextualize the trajectory bundle method, this section provides brief overviews of derivative-free optimization, trajectory optimization through sequential convex programming, and model predictive path integral control.

A. Derivative-Free Optimization

Derivative-Free Optimization (DFO) is a well-studied technique for solving optimization problems where derivatives are unavailable. John Dennis describes the DFO problem in [15] as "finding the deepest point of a muddy lake, given a boat and a plumb line, when there is a price to be paid for each sounding." With modern accelerators capable of massively parallel dynamics, cost, and constraint evaluation, there is still a price to take soundings, but we can now "buy in bulk." In a seminal 1965 paper, the Nelder-Mead method for derivative-free function minimization was proposed, where a simplex of sample points is used to approximate the cost landscape instead of derivatives. Methods like Mesh Adaptive Direct Search (MADS) [16] and NOMAD [17], [18] followed with improvements to the Nelder-Mead method.

Although there has been much work on general DFO [19], [20], there are two notably similar approaches to the trajectory bundle method. The first is the Constrained Optimization by Linear Interpolation (COBYLA) solver, where the cost and constraint functions are approximated with linear interpolation [15], and the second is the Gauss-Newton method of [21], where samples are used for linear interpolation. The trajectory bundle method builds on these two methods and specializes to trajectory optimization problems where the decision variables at different time steps are only coupled via the dynamics constraints. By exploiting this problem-specific structure and massively parallel simulation, the trajectory bundle method is a simple and robust method for solving trajectory optimization problems to tight constraint and optimality tolerances.

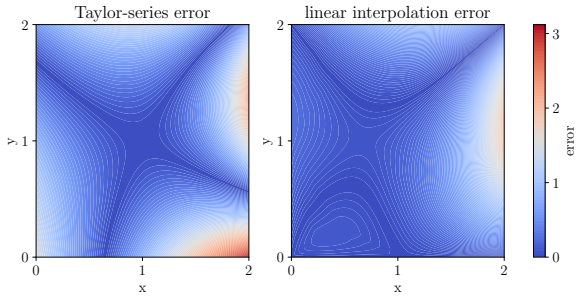

Fig. 2: A comparison of the accuracy of a first-order Taylor series taken about $(x, y) = (1, 1)$ with linear interpolation of the four corner points on the function $f(x, y) = \sin(x)e^y$. While the magnitudes of the errors are comparable between these two approximations, the patterns of these errors are notably different.

B. Trajectory Optimization through Sequential-Convex Programming

Trajectory optimization provides a rigorous and powerful mathematical framework for solving general optimal control problems as nonlinear programs. The cost and constraint functions can be arbitrary nonlinear functions that describe the task objective, laws of physics, and physical limits that must be enforced [22]. Assuming the cost and constraint functions are smooth and differentiable, this potentially nonlinear, non-convex problem can be solved by linearizing around a current iterate, forming a convex approximation of the original problem, and iterating until convergence. This method, known as Sequential Convex Programming (SCP), has been applied successfully to a wide variety of robotic systems such as rockets [23], orbital transfers [24], fixed-wing aircraft [25], rotorcraft [26], autonomous vehicles [27], and underwater vehicles [28].

The trajectory bundle method solves the trajectory optimization problem through sequential-convex programming similar to existing work, but differs in the way in which a convex approximation of the problem is formed. Instead of approximating the nonlinear cost and constraint functions by linearizing around the current iterate, we approximate the cost and constraint functions in a derivative-free manner by linearly interpolating samples within a trust region.

C. Model-Predictive Path Integral Control

Model-predictive path integral control (MPPI) is a sampling-based method for trajectory optimization or model-predictive control in which derivatives of the dynamics and cost functions are not required. The MPPI algorithm was first derived using a path-integral approach [29] then later re-derived from an information theoretic perspective [6], a stochastic search perspective [30], as well as through the use of mirror descent in an online learning context [7].

Given the rise of parallel computer architectures (GPUs and multi-threaded CPUs), MPPI has become a popular approach for tackling challenging real-time optimal control problems [31], [32] because the most expensive part of the algorithm — evaluating sampled rollouts and their correspond-

ing costs — can be done entirely in parallel. Additionally, MPPI can be extremely general and naturally amenable to black-box models learned from real-world data [6]. Despite its empirical success, MPPI performs poorly on open-loop unstable systems due to its single-shooting nature, and is unable to directly reason about constraints.

In this paper, we provide a new interpretation of MPPI as a sequential convex programming method and as a special case of the trajectory bundle method, and generalize sample-based optimal control to handle open-loop unstable systems and arbitrary constraints.

III. Background

Like many nonlinear optimization algorithms, the trajectory bundle method handles nonlinear cost and constraint functions by approximating them locally with affine functions. In this section, the standard method of approximating functions with a Taylor series is reviewed, followed by a derivative-free method using linear interpolation over sample points. Using these linear interpolants, the process for locally approximating a generic constrained optimization problem as convex is then presented.

A. Affine Function Approximation

An arbitrary function $p : \mathbf{R}^a \to \mathbf{R}^b$ is affine if it can be represented in the following form: $p_{\text{aff}}(y) = d + Cy$, where $d \in \mathbf{R}^b$ and $C \in \mathbf{R}^{b\times a}$. The process of locally approximating a nonlinear function with an affine function around a point $\bar{y}$ is often referred to as linearization, with $\bar{y}$ denoted as the linearization point. In this section, the standard method of approximation by first-order Taylor series is presented, followed by a derivative-free method that involves linear interpolation of sampled points.

1) Taylor Series: An affine approximation of this function $p(y) \approx \hat{p}(y)$ can be formed in the vicinity of an input value $\bar{y}$ through the use of the first-order Taylor series,

$$p(y) \approx \hat{p}(y) = p(\bar{y}) + \frac{\partial p}{\partial y}(y - \bar{y}), \tag{1}$$

where both the value and the Jacobian of p are calculated at the point $\bar{y}$. This approximation is exact at $\bar{y}$, and, generally speaking, becomes less accurate farther from $\bar{y}$.

2) Linear Interpolation: Alternatively, nonlinear functions can be approximated in a derivative-free manner by linearly interpolating between sampled function values. This is useful when the derivatives of a function are unavailable, challenging to compute, or unreliable.

For an affine function, any linear interpolation between two inputs is equal to the linear interpolation of the outputs. This means for an interpolation parameter $\theta \in [0, 1]$ and two inputs y_1 and y_2, the following holds:

$$p_{\text{aff}}(\underbrace{\theta y_1 + (1-\theta)y_2}_{\text{interpolated inputs}}) = \underbrace{\theta p_{\text{aff}}(y_1) + (1-\theta)p_{\text{aff}}(y_2)}_{\text{interpolated outputs}}. \tag{2}$$

This concept can be extended to m points with an interpolation vector $\alpha \in \mathbf{R}^m$ that belongs to a standard simplex:

$$\alpha \in \Delta^{m-1} = \left\{ \alpha \in \mathbf{R}^m \;\middle|\; \sum_{i=1}^{m} \alpha_i = 1,\, \alpha \geq 0 \right\}, \tag{3}$$

where again the convex combination of these m inputs is equal to the same convex combination of the m outputs,

$$p_{\text{aff}}\left(\sum_{i=1}^{m} \alpha_i y_i\right) = \sum_{i=1}^{m} \alpha_i p_{\text{aff}}(y_i). \tag{4}$$

This means that we can locally approximate the original nonlinear function p in the neighborhood of $\bar{y}$ by sampling m points from a distribution centered around $\bar{y}$ with $y_i \sim \mathcal{D}(\bar{y})$, and constraining the inputs to this approximation to be a linear combination of the sample points. For notational convenience, the lists of inputs and outputs are horizontally concatenated as columns of the matrices

$$W_y = \begin{bmatrix} y_1 & y_2 & \cdots & y_m \end{bmatrix} \in \mathbf{R}^{n_y \times m}, \tag{5}$$

$$W_p = \begin{bmatrix} p(z_1) & p(z_2) & \cdots & p(z_m) \end{bmatrix} \in \mathbf{R}^{n_r \times m}, \tag{6}$$

enabling the affine approximation $\hat{p}$ to be summarized as:

$$y = W_y \alpha \qquad \text{linear interpolation of inputs,} \tag{7}$$

$$\hat{p} = W_p \alpha \qquad \text{linear interpolation of outputs.} \tag{8}$$

We are effectively using the approximation $p(W_y\alpha) \approx W_p\alpha$, which is linear in α.

B. Approximation for Optimization

We will now consider a general nonlinear optimization problem and examine how these linearization techniques can be utilized to form a convex approximation of the original problem. This approach is used in sequential convex programming (SCP) methods where nonconvex optimization problems are solved by iteratively approximating the problem as convex in the neighborhood of the local iterate and solving for a step direction [33], [34], [22].

To demonstrate how an SCP method works with the two approximation techniques outlined, we examine a generic constrained optimization problem of the following form:

$$\begin{aligned} \underset{z}{\text{minimize}} \quad & \|r(z)\|_2^2 \\ \text{subject to} \quad & c(z) = 0, \end{aligned} \tag{9}$$

with a decision variable $z \in \mathbf{R}^{n_z}$, cost residual function $r : \mathbf{R}^{n_z} \to \mathbf{R}^{n_r}$, and constraint function $c : \mathbf{R}^{n_z} \to \mathbf{R}^{n_c}$. By approximating both of these functions as affine with a first-order Taylor series around a current iterate $\bar{z}$, we are left with the following convex optimization problem:

$$\begin{aligned} \underset{z}{\text{minimize}} \quad & \| \overbrace{r(\bar{z}) + \frac{\partial r}{\partial z}(z - \bar{z})}^{\hat{r}(z)} \|_2^2 \\ \text{subject to} \quad & \overbrace{c(\bar{z}) + \frac{\partial c}{\partial z}(z - \bar{z})}^{\hat{c}(z)} = 0. \end{aligned} \tag{10}$$

While this approximate problem is convex and we are guaranteed to find a globally optimal solution if one exists, we do

not have a guarantee of feasibility. There are circumstances in which the linearization of the constraint function results in infeasible approximate problems [34]. In order to guarantee that this problem is always feasible, many sequential convex programming methods convert the constraint into a penalty and reformulate (10) with an always-feasible variant,

$$\begin{aligned} \underset{z,\, s}{\text{minimize}} \quad & \overbrace{\| r(\bar{z}) + \frac{\partial r}{\partial z}(z - \bar{z}) \|_2^2}^{\hat{r}(z)} + \mu \|s\|_1 \\ \text{subject to} \quad & \overbrace{c(\bar{z}) + \frac{\partial c}{\partial z}(z - \bar{z})}^{\hat{c}(z)} + s = 0. \end{aligned} \tag{11}$$

where $\mu \in \mathbf{R}_+$ is a positive penalty weight and the ℓ_1-norm discourages constraint violations. No matter the structure of the cost and constraint functions, the convex optimization problem in (11) is guaranteed to always have a solution.

Alternatively, a linear interpolant such as that shown in (8) can be used to approximate the cost and constraint functions. To do this, m sample points centered around the current iterate $\bar{z}$ are used to evaluate the cost and constraint functions. These values are then horizontally concatenated into the following matrices:

$$W_z = \begin{bmatrix} z_1 & z_2 & \cdots & z_m \end{bmatrix} \in \mathbf{R}^{n_z \times m}, \tag{12}$$

$$W_r = \begin{bmatrix} r(z_1) & r(z_2) & \cdots & r(z_m) \end{bmatrix} \in \mathbf{R}^{n_r \times m}, \tag{13}$$

$$W_c = \begin{bmatrix} c(z_1) & c(z_2) & \cdots & c(z_m) \end{bmatrix} \in \mathbf{R}^{n_c \times m}. \tag{14}$$

The interpolation vector $\alpha \in \mathbf{R}^m$ is used to interpolate between these samples and their corresponding cost and constraint values. These approximations are used to recreate the relaxed problem shown in (11) as the following:

$$\begin{aligned} \underset{\alpha,\, s}{\text{minimize}} \quad & \| \overbrace{W_r \alpha}^{\hat{r}(z)} \|_2^2 + \mu \|s\|_1 \\ \text{subject to} \quad & \overbrace{W_c \alpha}^{\hat{c}(z)} + s = 0, \\ & \alpha \in \Delta^{m-1}, \end{aligned} \tag{15}$$

where the optimal solution is $z^* = W_z \alpha^*$. It is important to note the similarities between (11) and (15), where the only difference is the method to approximate the cost, residual, and constraint functions. Another key difference between these methods is the implicit trust region present in the simplex constraint on α. Since $\alpha \in \Delta^{m-1}$, the solution is restricted to the convex hull of the sample points. Using a sampling scheme that only samples points within a set trust region, the solution to (15) is guaranteed to stay within the trust region.

IV. The Trajectory Bundle Method

In this section, we outline a canonical trajectory optimization problem and approximate the cost and constraint functions using linearly interpolated trajectory bundles.

Trajectories will be represented in discrete time as a list of vectors. For a dynamical system with a state $x \in \mathbf{R}^{n_x}$ and control $u \in \mathbf{R}^{n_u}$, the discrete-time dynamics function

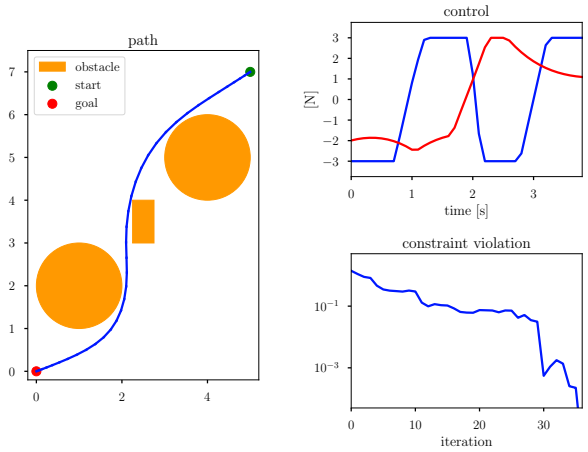

Fig. 3: A double integrator with acceleration control is tasked with navigating around three obstacles to a goal position. The trajectory bundle method can directly reason about these nonlinear, non-convex constraints without derivatives, with strong constraint satisfaction and optimality achieved in fewer than 40 iterations.

$x^{(k+1)} = f(x^{(k)}, u^{(k)})$ maps the state and control at time-step k to the state at $k+1$. A trajectory comprised of N time steps is represented by $(x^{(1:N)}, u^{(1:N-1)})$, such that numerical optimization can be used to solve for these values.

A. Trajectory Optimization

A canonical trajectory optimization problem considering a trajectory with N time steps is represented as:

$$\begin{aligned} \underset{x^{(1:N)},\, u^{(1:N-1)}}{\text{minimize}} \quad & \|r_N(x^{(N)})\|_2^2 + \sum_{k=1}^{N-1} \|r_k(x^{(k)}, u^{(k)})\|_2^2 \\ \text{subject to} \quad & x^{(k+1)} = f(x^{(k)}, u^{(k)}), \\ & c(x^{(k)}, u^{(k)}) \geq 0, \end{aligned} \tag{16}$$

where $c(x^{(k)}, u^{(k)})$ is a generic constraint function. We will assume that all relevant constraints are expressed in this form, including initial and goal constraints, state and control limits, and other general stage-wise constraints.

The problem format in (16) is often referred to as multiple shooting [35], where both the state and control histories are optimized over, and the trajectory only becomes dynamically feasible at convergence. This differs from single shooting, where only the controls are optimized over, and a rollout is performed to recover the states. One important distinction between these two methods is that in single shooting, the discrete-time dynamics must be evaluated sequentially $N-1$ times during the rollout, while in multiple shooting, the $N-1$ dynamics constraints can be evaluated entirely in parallel. This is especially relevant with GPU-based physics simulation, where the speed of a single simulation can be comparable to thousands of simulations run in parallel.

B. Solving Multiple Shooting with Trajectory Bundles

The trajectory bundle method is able to reason about the trajectory optimization problem in (16) without having

to differentiate any of the cost, dynamics, or constraint functions. Instead of using derivatives to approximate these functions with their first-order Taylor series, sampled trajectories near the current iterate are used to evaluate these functions for approximation with linear interpolation. This idea is shown in III-A Given an initial guess or current iterate $(\bar{x}^{(1:N)}, \bar{u}^{(1:N-1)})$, the costs, constraints, and dynamics functions are computed for each of the M samples surrounding each knot points. To demonstrate this, let us examine a single knot point, k, where the current iterate is $(\bar{x}^{(k)}, \bar{u}^{(k)})$. From here, m points are sampled near the iterate, and these samples are horizontally concatenated into the following matrices:

$$W_x^{(k)} = \begin{bmatrix} x_1^{(k)} & x_2^{(k)} & \cdots & x_m^{(k)} \end{bmatrix} \in \mathbf{R}^{n_x \times m}, \tag{17}$$

$$W_u^{(k)} = \begin{bmatrix} u_1^{(k)} & u_2^{(k)} & \cdots & u_m^{(k)} \end{bmatrix} \in \mathbf{R}^{n_u \times m}, \tag{18}$$

after which, all of the cost, dynamics, and constraint functions are computed and stored in a similar fashion,

$$W_r^{(k)} = \begin{bmatrix} r_1^{(k)} & r_2^{(k)} & \cdots & r_m^{(k)} \end{bmatrix} \in \mathbf{R}^{n_r \times m}, \tag{19}$$

$$W_f^{(k)} = \begin{bmatrix} f_1^{(k)} & f_2^{(k)} & \cdots & f_m^{(k)} \end{bmatrix} \in \mathbf{R}^{n_x \times m}, \tag{20}$$

$$W_c^{(k)} = \begin{bmatrix} c_1^{(k)} & c_2^{(k)} & \cdots & c_m^{(k)} \end{bmatrix} \in \mathbf{R}^{n_c \times m}, \tag{21}$$

where $r_i^{(k)} = r(x_i^{(k)}, u_i^{(k)})$, $f_i^{(k)} = f(x_i^{(k)}, u_i^{(k)})$, and $c_i^{(k)} = c(x_i^{(k)}, u_i^{(k)})$. These matrices are computed for time-steps $1 \to N$, with time-step N. Together, these matrices can be used to locally approximate the potentially nonconvex optimization problem in (16) as the following convex optimization problem:

$$\begin{aligned} \underset{\alpha^{(1:N)}}{\text{minimize}} \quad & \| \overbrace{W_r^{(N)}\alpha^{(N)}}^{\hat{r}_N(x_N)} \|_2^2 + \sum_{k=1}^{N-1} \| \overbrace{W_r^{(k)}\alpha^{(k)}}^{\hat{r}_k(x_k,u_k)} \|_2^2 \\ & + \mu \sum_{k=1}^{N-1} \|s^{(k)}\|_1 + \|w^{(k)}\|_1 \\ \text{subject to} \quad & \overbrace{W_x^{(k+1)}\alpha^{(k+1)}}^{x^{(k+1)}} = \overbrace{W_f^{(k)}\alpha^{(k)}}^{\hat{f}(x^{(k)},u^{(k)})} + s^{(k)}, \\ & \overbrace{W_c^{(k)}\alpha^{(k)}}^{\hat{c}(x_k,u_k)} + w_k \geq 0, \\ & \alpha^{(k)} \in \Delta^{m-1}, \end{aligned} \tag{22}$$

where s and w are slack variables that, analogous to (11), ensure the problem remains feasible.

Each time this problem is solved, the new iterates $(x^{(1:N)}, u^{(1:N-1)})$ are used to generate m new samples, and the problem is formed and solved again. This SCP-based algorithm repeats until convergence which, in this particular case, is synonymous with constraint satisfaction.

C. MPPI as a Trajectory Bundle Problem

In MPPI, control policies are sampled and used to generate simulated rollouts with associated costs, and a weighted average based on these costs is used to "blend" the sampled policies, with this process at each controller call, the nominal control policy is converging to local optimality.

A single iteration of MPPI starts with m sampled control policies $U^{(1:m)}$ from a distribution centered around a nominal policy $\bar{U}$. Each of these samples is used in a forward dynamics rollout from an initial condition x_0 and an associated cost J_i is computed using the rollout from sample i. Using the costs from these rollouts, $J_{1:m} \in \mathbf{R}^m$, weights $\alpha \in \mathbf{R}^m$ are computed with the softmax function:

$$\alpha_i = \frac{e^{-\frac{J_i}{\lambda}}}{\sum_{i=1}^m e^{-\frac{J_i}{\lambda}}}, \tag{23}$$

where λ is a non-negative temperature parameter. Using these weights, the resulting updated control policy is a weighted average of the samples, computed as

$$U = \sum_{i=1}^m \alpha_i U^{(i)} \tag{24}$$

In the limit $\lambda \to 0$, the MPPI update selects the single best control sequence among the samples, which is also known as predictive sampling [36].

The MPPI update rule can also be derived as a special case of the trajectory bundle method, where the trajectory is represented with single shooting and there are, therefore, no explicit dynamics constraints. Given the m control samples and associated costs, a convex optimization problem solving for the convex combination of trajectories that minimizes the interpolated cost along with a negative entropy regularizer is the following:

$$\begin{aligned} \underset{\alpha}{\text{minimize}} \quad & \overbrace{w_J^T \alpha}^{\hat{J}(u)} - \overbrace{\lambda \sum_{i=1}^m \alpha_i \log \alpha_i}^{\text{entropy regularization}} \\ \text{subject to} \quad & \alpha \in \Delta^{m-1}, \end{aligned} \tag{25}$$

where λ is the regularization parameter, and the convexity of the negative entropy term makes this a convex optimization problem. The solution to this problem can be computed in closed form as (23). Just like in MPPI, $\lambda \to 0$ corresponds to the unregularized bundle problem, where the solution is simply the best sample as no linear combination of the samples can produce a lower cost. This interpretation of MPPI gives a new perspective through the lens of convex optimization, which can both provide a deeper understanding of the algorithm and provide opportunities for future work.

D. Sampling Strategies

We implement multiple sampling strategies, including Gaussian and uniform distributions, but found that performance was largely independent of the distribution. The experiments in the paper use a simple deterministic coordinate-wise perturbation scheme:

$$z_i = \bar{z} + e_i \Delta z_i \qquad \forall i \in [1, n] \tag{26}$$

$$z_{i+n} = \bar{z} - e_i \Delta z_i \qquad \forall i \in [1, n] \tag{27}$$

where $\bar{z}$ is the current iterate, Δz defines the trust region, and e_i is the unit vector in the i-th coordinate direction, and $z_{2n+1} = \bar{z}$. We leave detailed analysis on sampling strategies for future work.

E. Convergence Behavior

While we defer a formal convergence analysis of the trajectory bundle method to future work, we note that the convergence properties of notable special cases of our method been extensively studied [37], [15]. In the single-shooting setting with entropy regularization, as noted in Sec. IV-C, our method coincides with MPPI, which has a well developed convergence theory [6]. In the multiple-shooting setting, our method is similar to existing gradient-free trust-region SQP methods, and therefore should enjoy similar convergence behavior [38], [39].

V. Experiments and Results

In this section, we solve several motion planning and control problems using the trajectory bundle method. In a unified framework, it addresses challenging problems typically associated with either derivative-based SCP (Sec. V-A, V-B, V-D) or derivative-free sampling-based methods (Sec. V-D). We also show that it combines the strengths of both by solving a class of problems neither method alone can handle (Sec. V-C).

Unless otherwise noted, the convex optimization problems are solved with Clarabel [40] through CVXPY [41] and we consider TBM to be converged when the maximum constraint violation is below 10^{-4}. An open-source implementation of the solver and experiments will be made available upon publication.

Features	SCP	MPPI	TBM (ours)
State constraints	✓	✗	✓
Reasoning over long horizons	✓	✗	✓
Non-differentiable costs and constraints	✗	✓	✓
Black-box models	✗	✓	✓
Unstable systems	✓	✗	✓

TABLE I: Comparison of different trajectory optimization methods.

A. Double Integrator Collision Avoidance

To demonstrate the ability of the trajectory bundle method to handle nonlinear/nonconvex constraints, we consider the collision avoidance example in Fig. 3. An acceleration-limited double integrator ($u = \ddot{x}$) must reach the goal while avoiding obstacles. Despite lacking derivative information for these nonconvex constraints, the method converges to a feasible collision-free trajectory in fewer than 40 iterations.

B. Quatrotor Figure Eight Tracking

Next, a quadrotor with rotor-velocity control is tasked with tracking a skewed figure eight path through space over a five-second horizon. The trajectory is discretized into 100 time-steps, and the resulting optimal trajectory smoothly tracks this aggressive reference while maintaining a smooth control commands. The angular velocity of the quadrotor can reach over 200 degrees per second, where the attitude dynamics are highly nonlinear. In fewer than 60 iterations, the trajectory bundle method can solve this problem to the given constraint tolerance. In comparison, even after millions of simulation steps, MPPI fails to stabilize this open-loop unstable system over a long horizon, resulting in an unrecoverable crash.

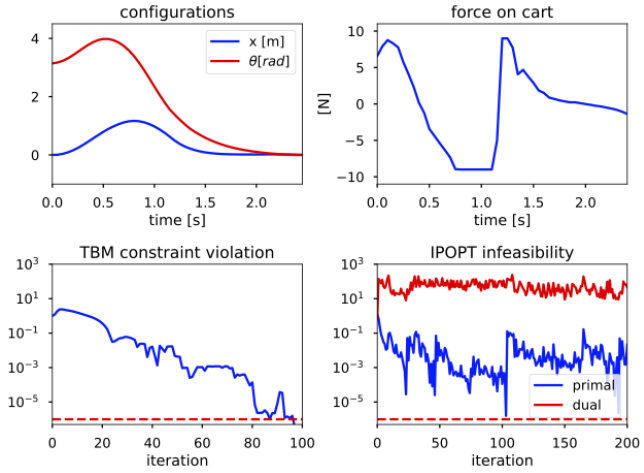

Fig. 4: The cartpole swingup task with a neural-network dynamics model. TBM (ours) finds smooth state (top left) and control (top right) trajectories and converges to tight (10^{-6}, red dashed line) constraint satisfaction (bottom left). The baseline IPOPT solver fails to converge (bottom right).

The problems in Sec. V-A and V-B are highly nonlinear/non-convex, highly constrained, and deal with long trajectories. These all present challenges to single-shooting methods like MPPI due to unstable rollouts, a problem noticeably absent from multiple shooting formulations.

C. Cartpole Swingup with a Neural Dynamics Model

Many robotics simulators are unable to produce smooth and reliable derivatives; this can be a result of nonsmooth impact events, but also if the dynamics are represented with a nonsmooth neural network. In the latter case, it is not uncommon for a learned dynamics model to match the values of the real model well, but not the derivatives. In Fig. 4, we solve the canonical cartpole swingup trajectory optimization problem with a neural network dynamics model. In this problem, a horizontal cart with an attached pole must swing itself from its stable equilibrium at the bottom to the unstable equilibrium at the top in 2.5 seconds. The problem is discretized into 50 time steps. Control bounds and goal constraints are also applied. Using simulated MuJoCo [14] dynamics data, we train a 2-layer MLP with 64 units per layer and ReLU activation functions to predict the next robot state $x^{(k+1)}$ given the current state $x^{(k)}$ and control $u^{(k)}$. Despite the discontinuous MLP dynamics, TBM successfully solves the problem to tight (10^{-6}) tolerance in under 100 iterations. On the other hand, IPOPT [42], a standard nonlinear optimization solver relying on derivative information, fails to converge to an optimal solution even after $10,000$ iterations due to the nonsmoothness of the ReLU network. We only show the first 200 iterations in Fig. 4 for visual clarity.

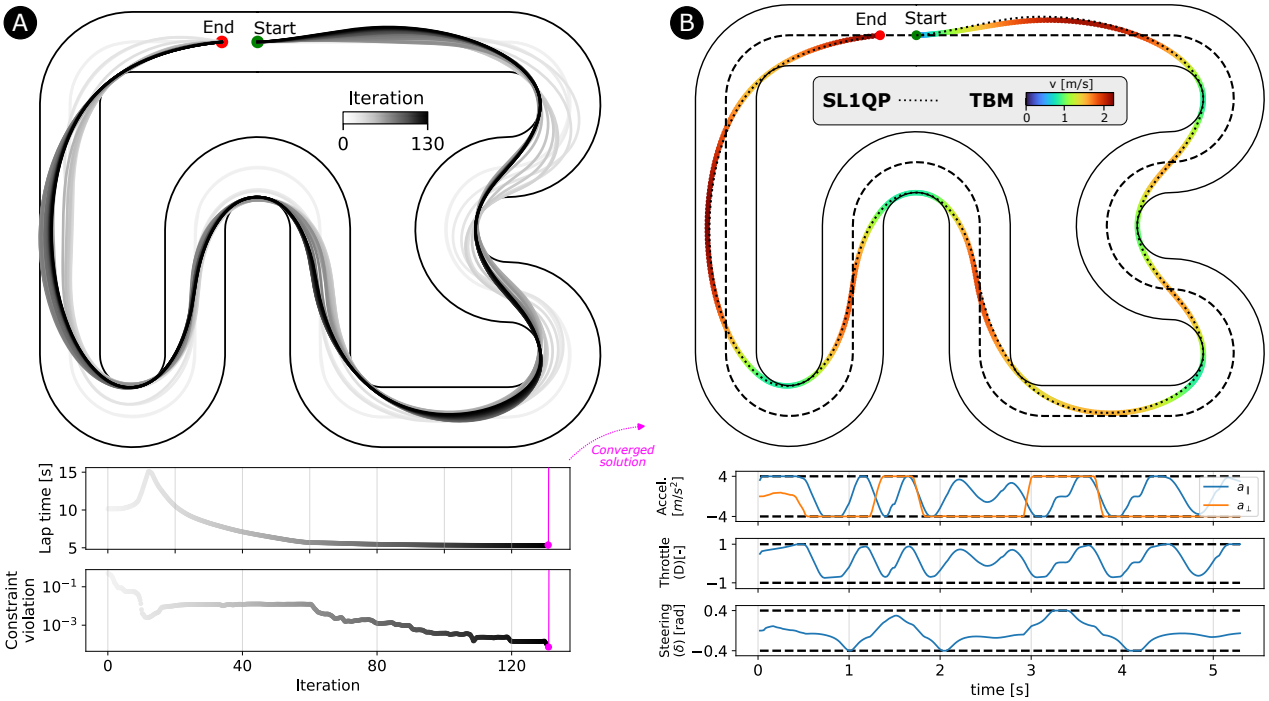

Fig. 5: Minimum-time trajectories using TBM with multiple shooting for a 1:43 autonomous racing example. This demonstrates that TBM (velocity color-coded) and SL1QP (a standard SCP gradient-based method, dotted) produce identical trajectories. *Panel A*: Trajectories progress from centerline initialization (iteration 0) to convergence (iteration 130). The upper section displays the racetrack and trajectories; the lower rows show lap times and constraint violations versus iteration, with grayscale gradients indicating progress. *Panel B*: Converged trajectory at iteration 130. The upper plot shows that the TBM and the SL1QP racelines are the same; the lower section depicts TBM's lateral/longitudinal acceleration, throttle, and steering over time, with dashed lines indicated bounds.

While sample-based MPPI naturally incorporates potentially nonsmooth learned black-box models, it cannot reason over long horizons or about the general state and goal constraints required in many robotics problems like those described in Sec. V-A, V-B, and V-C. As a derivative-free trajectory optimization method, TBM is also amendable to these black-box dynamics models and can handle general (potentially black-box) costs and constraints.

D. Race Car Min-Time Optimization and Local Planning

To demonstrate TBM's versatility in handling complex dynamical systems, we tested it on a 1:43 scale autonomous racing car with nonlinear constraints. The model uses a state vector representing position, yaw angle, velocity, throttle, and steering angle, with constraints on throttle, steering angle, and accelerations. We conducted two experiments: First, an offline trajectory optimization minimizing lap time, where TBM produced a solution equivalent to SL1QP (a gradient-based method) with only a 5ms difference in lap time. Second, we created a local planning scenario with obstacles requiring real-time avoidance, where TBM with entropy regularization produced results identical to MPPI. These results confirm that TBM offers a unified framework for solving problems traditionally handled by separate algorithms, matching SQP's for constrained offline planning and MPPI for online reactive control.

VI. Conclusions

In this work, we present the trajectory bundle method, a derivative-free trajectory optimization technique capable of solving nonconvex constrained optimization problems with strong constraint satisfaction. Instead of approximating the nonconvex functions with first-order Taylor series, the trajectory bundle method samples points locally and computes the cost, dynamics, and constraint functions for each of these samples in what we refer to as bundles. These bundles are used to linearly interpolate between these sampled values to approximate the cost, dynamics, and constraint functions. After the computation of these highly parallelizeable function calls, a convex optimization problem is solved where the nonconvex functions are replaced with linear interpolants, and the solution is used to generate new samples for the bundles. The effectiveness of this method is demonstrated on a variety of robotics platforms.

While the trajectory bundle method is a flexible and capable framework for solving trajectory optimization problems, it is not without limitations. Firstly, while you can readily compute constraint violations for these problems, a reliable metric for optimality is still an open question since derivatives are required to compute optimality conditions. Another challenge has been identifying the ideal distribution to draw samples from, while uniform and Gaussian distributions have been sufficient for the examples shown in this paper, there are

certainly opportunities to use more expressive and potentially learned distributions for better convergence. The last is a robust convergence rate guarantee, one that will likely require an adaptive penalty μ and adaptation of the trust region. We leave these challenges open for future work in the area.

REFERENCES

[1] R. E. Kalman. A New Approach to Linear Filtering and Prediction Problems. *Journal of Basic Engineering*, 82(1):35–45, March 1960.

[2] Francesco Borrelli, Alberto Bemporad, and Manfred Morari. *Predictive Control for Linear and Hybrid Systems*. Cambridge University Press, 2017.

[3] Stephen Boyd and Lieven Vandenberghe. *Convex Optimization*. Cambridge University Press, 2004.

[4] Jean-Jacques Slotine and Weiping Li. *Applied Nonlinear Control*. Prentice Hall, Engelwood Cliffs, NJ, 1991.

[5] Roy Featherstone. *Robot Dynamics Algorithms*. The Springer International Series in Engineering and Computer Science. Springer US, 1987.

[6] Grady Williams, Paul Drews, Brian Goldfain, James M. Rehg, and Evangelos A. Theodorou. Information-Theoretic Model Predictive Control: Theory and Applications to Autonomous Driving. *IEEE Transactions on Robotics*, 34(6):1603–1622, December 2018.

[7] Nolan Wagener, Ching-An Cheng, Jacob Sacks, and Byron Boots. An Online Learning Approach to Model Predictive Control, October 2019.

[8] C Daniel Freeman, Anton Raichuk, Sertan Girgin, Erik Frey, Igor Mordatch, and Olivier Bachem. Brax - A Differentiable Physics Engine for Large Scale Rigid Body Simulation. In *NeurIPS*, page 13, New Orleans, December 2021.

[9] Kevin Tracy, Taylor A. Howell, and Zachary Manchester. Differentiable Collision Detection for a Set of Convex Primitives. In *2023 IEEE International Conference on Robotics and Automation (ICRA)*, pages 3663–3670, London, United Kingdom, May 2023. IEEE.

[10] H. J. Terry Suh, Max Simchowitz, Kaiqing Zhang, and Russ Tedrake. Do Differentiable Simulators Give Better Policy Gradients?, August 2022.

[11] Taylor A. Howell, Simon Le Cleac'h, J. Zico Kolter, Mac Schwager, and Zachary Manchester. Dojo: A Differentiable Physics Engine for Robotics. *http://arxiv.org/abs/2203.00806*, June 2022.

[12] Viktor Makoviychuk, Lukasz Wawrzyniak, Yunrong Guo, Michelle Lu, Kier Storey, Miles Macklin, David Hoeller, Nikita Rudin, Arthur Allshire, Ankur Handa, and Gavriel State. Isaac Gym: High Performance GPU-Based Physics Simulation For Robot Learning, August 2021.

[13] Mayank Mittal, Calvin Yu, Qinxi Yu, Jingzhou Liu, Nikita Rudin, David Hoeller, Jia Lin Yuan, Pooria Poorsarvi Tehrani, Ritvik Singh, Yunrong Guo, Hammad Mazhar, Ajay Mandlekar, Buck Babich, Gavriel State, Marco Hutter, and Animesh Garg. ORBIT: A Unified Simulation Framework for Interactive Robot Learning Environments, January 2023.

[14] E. Todorov, T. Erez, and Y. Tassa. MuJoCo: A physics engine for model-based control. In *2012 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 5026–5033, October 2012.

[15] M. J. D. Powell. A Direct Search Optimization Method That Models the Objective and Constraint Functions by Linear Interpolation. In Susana Gomez and Jean-Pierre Hennart, editors, *Advances in Optimization and Numerical Analysis*, pages 51–67. Springer Netherlands, Dordrecht, 1994.

[16] Charles Audet and J. E. Dennis. Mesh Adaptive Direct Search Algorithms for Constrained Optimization. *SIAM Journal on Optimization*, 17(1):188–217, January 2006.

[17] Sébastien Le Digabel. Algorithm 909: NOMAD: Nonlinear Optimization with the MADS Algorithm. *ACM Trans. Math. Softw.*, 37(4):44:1–44:15, February 2011.

[18] Charles Audet, Sébastien Le Digabel, Viviane Rochon Montplaisir, and Christophe Tribes. NOMAD version 4: Nonlinear optimization with the MADS algorithm, May 2021.

[19] Mykel J. Kochenderfer and Tim A. Wheeler. *Algorithms for Optimization*. The MIT Press, Cambridge, Massachusetts, 2019.

[20] A. R. Conn, Katya Scheinberg, and Luís N. Vicente. *Introduction to Derivative-Free Optimization*. Number 8 in MPS-SIAM Series on Optimization. SIAM, Philadelphia, Pa, 2009.

[21] Coralia Cartis and Lindon Roberts. A derivative-free Gauss–Newton method. *Mathematical Programming Computation*, 11(4):631–674, December 2019.

[22] Danylo Malyuta, Taylor P. Reynolds, Michael Szmuk, Thomas Lew, Riccardo Bonalli, Marco Pavone, and Behcet Acikmese. Convex Optimization for Trajectory Generation. *arXiv:2106.09125 [cs, eess, math]*, June 2021.

[23] Lars Blackmore. Autonomous Precision Landing of Space Rockets. *The Bridge*, 4(46):15–20, 2016.

[24] James D Thorne and Christopher D Hall. Minimum-Time Continuous-Thrust Orbit Transfers Using the Kustaanheimo-Stiefel Transformation. 20(4):3.

[25] Rick Cory and Russ Tedrake. Experiments in Fixed-Wing UAV Perching. In *AIAA Guidance, Navigation, and Control Conference*, Honolulu, Hawaii, 2008.

[26] Sihao Sun, Angel Romero, Philipp Foehn, Elia Kaufmann, and Davide Scaramuzza. A Comparative Study of Nonlinear MPC and Differential-Flatness-Based Control for Quadrotor Agile Flight. *IEEE Transactions on Robotics*, pages 1–17, 2022.

[27] Jianyu Chen, Yutaka Shimizu, Liting Sun, Masayoshi Tomizuka, and Wei Zhan. Constrained Iterative LQG for Real-Time Chance-Constrained Gaussian Belief Space Planning. In *2021 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 5801–5808, September 2021.

[28] Giorgos Mamakoukas, Malcolm A. MacIver, and Todd D. Murphey. Sequential action control for models of underactuated underwater vehicles in a planar ideal fluid. In *2016 American Control Conference (ACC)*, pages 4500–4506, 2016.

[29] Grady Williams, Paul Drews, Brian Goldfain, James M. Rehg, and Evangelos A. Theodorou. Aggressive driving with model predictive path integral control. In *2016 IEEE International Conference on Robotics and Automation (ICRA)*, pages 1433–1440, May 2016.

[30] Ziyi Wang, Oswin So, Keuntaek Lee, and Evangelos A. Theodorou. Adaptive Risk Sensitive Model Predictive Control with Stochastic Search. In *Proceedings of the 3rd Conference on Learning for Dynamics and Control*, pages 510–522. PMLR, May 2021.

[31] Juan Alvarez-Padilla, John Z. Zhang, Sofia Kwok, John M. Dolan, and Zachary Manchester. Real-time whole-body control of legged robots with model-predictive path integral control. *arXiv preprint arXiv:2409.10469*, 2024.

[32] Albert H. Li, Preston Culbertson, Vince Kurtz, and Aaron D. Ames. DROP: Dexterous Reorientation via Online Planning, October 2024.

[33] Philip E. Gill, Walter Murray, and Michael A. Saunders. SNOPT: An SQP Algorithm for Large-Scale Constrained Optimization. *SIAM Review*, 47(1):99–131, January 2005.

[34] Jorge Nocedal and Stephen J. Wright. *Numerical Optimization*. Springer, 2nd edition, 2006.

[35] C R Hargraves and S W Paris. Direct Trajectory Optimization Using Nonlinear Programming and Collocation. *J. Guidance*, 10(4):338–342, 1987.

[36] Taylor Howell, Nimrod Gileadi, Saran Tunyasuvunakool, Kevin Zakka, Tom Erez, and Yuval Tassa. Predictive sampling: Real-time behaviour synthesis with mujoco. *arXiv preprint arXiv:2212.00541*, dec 2022.

[37] Charles Audet and Warren Hare. *Derivative-Free and Blackbox Optimization*. Springer Series in Operations Research and Financial Engineering. Springer International Publishing, Cham, 2017.

[38] Roger Fletcher, Nicholas IM Gould, Sven Leyffer, Philippe L Toint, and Andreas Wächter. Global convergence of a trust-region sqp-filter algorithm for general nonlinear programming. *SIAM Journal on Optimization*, 13(3):635–659, 2002.

[39] Stephen J Wright and Matthew J Tenny. A feasible trust-region sequential quadratic programming algorithm. *SIAM journal on optimization*, 14(4):1074–1105, 2004.

[40] Paul J. Goulart and Yuwen Chen. Clarabel: An interior-point solver for conic programs with quadratic objectives, May 2024.

[41] Steven Diamond and Stephen Boyd. CVXPY: A Python-embedded modeling language for convex optimization. *Journal of Machine Learning Research*, 17(83):1–5, 2016.

[42] 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(1):25–57, March 2006.