Improved Approximation Accuracy for Nonconvex Trajectory Optimization via Trajectory Sensitivities

Xiaofei Wu Michael W. Fisher Stephen L. Smith

***Abstract*— Trajectory optimization is valuable for a wide range of applications, from motion planning for mobile robots, to aircraft flight planning. However, nonlinear dynamic models lead to challenging nonconvex trajectory optimization problems. Many existing approaches formulate them as multistage programs and rely on derivatives of each stage to obtain a local approximation at each iteration, in which case quality of approximation when solving the optimization program has significant impact on convergence behavior. In this work, we develop a novel approach for obtaining improved local approximations when solving nonconvex trajectory optimization problems. By performing an input-to-state reformulation of system dynamics, we use trajectory sensitivities, which are derivatives of the entire system trajectory with respect to control inputs, to form local approximations. Local convergence guarantees for the proposed method are presented. The method is applied to generate trajectories for an autonomous vehicle, and is extended to include scenario with static obstacles. Simulation on a variety of reference paths show that the proposed method outperforms the traditional Sequential Quadratic Programming (SQP) in terms of local approximation accuracy, allowable trust-region radius, iterations to converge, and total solver time, and is less prone to failure when handling multiple obstacles with a complex reference.**

I. Introduction

Trajectory optimization is useful for many applications, such as motion planing and control of mobile robots, autonomous vehicles, and aircraft [1]–[4]. In this paper we study trajectory optimization for a control system subject to constraints. The optimality of a trajectory can be measured using predefined cost indices and is application dependent. Examples of constraints include feasibility according to the system dynamics, control input constraints, and collision avoidance constraints.

By considering a discrete time state space system model and optimizing the trajectory over a finite time horizon, the trajectory optimization problem is converted to a nonlinear programming problem (NLP) [5]. However, for many applications, nonlinear system dynamics are needed to improve accuracy of trajectories, which results in complex nonconvex optimization problems. To solve these problems, line-search and trust-region methods are two commonly-used approaches. Examples of line-search methods include Quasi-Newton methods and interior point methods that are implemented by the widely used solvers IPOPT and ForcesPro [6]–[8]. Trust region methods, such as Sequential Quadratic Programming (SQP), approximate and solve the original NLP while defining a trust region to control the accuracy of approximations [9] [10].

In this paper, we propose an input-to-state reformulation of the nonconvex trajectory optimization problem, which can be solved using SQP with trajectory-sensitivities-based constraint linearization. To do so, dynamics constraints of the system are rewritten as a map from all control inputs to the full trajectory of system states. Then, SQP is used to solve the resulting problem. When constructing the quadratic approximation of the original problem, existing approaches typically rely on derivatives of each stage to effectively obtain a local approximation of the problem at each iteration [6]. However, poor approximation quality can lead to reduced convergence rates, or even failure to converge. We propose to use trajectory sensitivities to perform linearization of the constraints with improved approximation accuracy when solving the optimization program.

Trajectory sensitivities are the derivatives of the system trajectory with respect to all control inputs [11] [12]. Unlike derivatives with respect to states and inputs at each stage, trajectory sensitivities consider the initial state and the entire input trajectories, and do not depend on other states. Thus, trajectory-sensitivities-based approximation does not suffer from accumulation of numerical error of multistage computation. Using trajectory sensitivities improves the numerical performance of the optimization algorithm, enables faster convergence and reduces the possibility of failure.

The proposed approach is used to generate trajectories for an autonomous vehicle, and is compared to the traditional SQP method. With a single-track bicycle model of vehicle being used, the resulting trajectory optimization problem is nonconvex. We show that with Dubin's reference paths, our method allows the use of larger trust region radii due to the improved approximation accuracy using trajectory sensitivities, and thus converges in fewer iterations and less solver time. We also show that our method is less prone to convergence failure with a path smoothing example with obstacle avoidance, as the collision avoidance constraints are approximated more accurately in the optimization problem.

The contributions of this work are as follows: (i) we propose a novel reformulation of nonconvex trajectory optimization problems by using an input-to-state map, (ii) we develop an efficient SQP-based approach to solve the reformulated problem, which uses trajectory sensitivities to construct local quadratic approximation of the original problem, (iii) we provide a local convergence guarantees for the proposed

This research is partially supported by the Natural Sciences and Engineering Research Council of Canada (NSERC)

The authors are with the Department of Electrical and Computer Engineering, University of Waterloo, Waterloo, ON, N2L 3G1 Canada (x334wu@uwaterloo.ca; michael.fisher@uwaterloo.ca; stephen.smith@uwaterloo.ca)

method, and (iv) we test our approach by applying it to generate trajectories for an autonomous vehicle, and compare the results with the traditional SQP method to capture the impact of trajectory sensitivities.

II. Preliminaries

Consider a discrete-time trajectory optimization problem, which is formulated as an NLP over a finite time horizon subject to equality and inequality constraints

$$\begin{aligned} \min_{x^k, u^k} \quad & J(\mathbf{x}, \mathbf{u}) \\ \text{s.t.} \quad & x^{k+1} = g(x^k, u^k), \ \forall\, k \in \{0, 1, ..., T-1\} \\ & h(x^k, u^k) \leq 0, \qquad \forall\, k \in \{0, 1, ..., T-1\} \\ & \mathbf{x} = \left[x_1^\intercal, ..., x_T^\intercal\right]^\intercal, \ \mathbf{u} = \left[u_1^\intercal, ..., u_T^\intercal\right]^\intercal, \end{aligned} \tag{1}$$

where $x^k \in \mathbb{R}^n$ is the state at time k; $u^k \in \mathbb{R}^m$ is the input at time k; T is the length of the time horizon; $\mathbf{x} \in \mathbb{R}^{nT}$ and $\mathbf{u} \in \mathbb{R}^{mT}$ are the collections of all states and inputs over the time horizon T, respectively; $J : \mathbb{R}^{nT \times mT} \to \mathbb{R}$ is the objective function; $g : \mathbb{R}^{n \times m} \to \mathbb{R}^n$ is the nonlinear system dynamics; and $h : \mathbb{R}^{n \times m} \to \mathbb{R}^p$ is the nonlinear element-wise inequality constraints.

In this paper, we consider the setting where the objective function J is quadratic, with which many practical objectives can be formulated with. We assume that the constraints g and h are C^2, and in general they are nonlinear and nonconvex.

A. Sequential Quadratic Programming

To solve the NLP in (1), we adopt an SQP based approach, which is a common approach to trajectory optimization. Being a trust region method, SQP allows us to use the size of the trust region as a metric for quantifying and comparing local approximation accuracy. In SQP, we iteratively solve a series of quadratic approximations of the original problem until the solution converges. Conventionally, the quadratic approximations are formulated by linearizing the constraints [6]. Take equality constraints in Equation (1) as an example, they are expressed as

$$x_{s+1}^{k+1} = x_s^{k+1} + \frac{\partial g}{\partial x}(x_{s+1}^k - x_s^k) + \frac{\partial g}{\partial u}(u_{s+1}^k - u_s^k)$$

for every time step k and each iteration s. The multistage nature of these linearizations often lead to accumulation of numerical error as k grows larger.

B. Trajectory Sensitivity

Trajectory sensitivities are the derivatives of the system state with respect to the input trajectory at each time step. Using system dynamics in (1), trajectory sensitivity at time step $k+1$ can be efficiently computed as

$$\left.\frac{dx}{d\mathbf{u}}\right|_{x^{k+1}} = \left.\frac{\partial g}{\partial x}\frac{dx}{d\mathbf{u}}\right|_{x^k} + \left.\frac{\partial g}{\partial \mathbf{u}}\right|_{u^k}. \tag{2}$$

The full trajectory sensitivities are evaluated in a recursive fashion. With this, the derivatives of inequality constraints can be computed

$$\left.\frac{dh}{d\mathbf{u}}\right|_{x^{k+1}} = \left.\frac{\partial h}{\partial x}\frac{dx}{d\mathbf{u}}\right|_{x^k} + \left.\frac{\partial h}{\partial \mathbf{u}}\right|_{u^k}. \tag{3}$$

C. Others

A matrix-induced inner product is defined as $< \mathbf{x}, \mathbf{x} >_A = \mathbf{x}^\intercal A \mathbf{x}$ where A is some matrix, and can be used to measure distance. We use the symbol $\|\cdot\|$ to represent a norm, and the symbols $B_r(p)$ and $\overline{B}_r(p)$ to denote the ball and the closed ball of radius r around point p, respectively.

III. Method

A. NLP Reformulation

First, we perform an input-to-state reformulation of the problem in (1). Given some initial state x^0, all subsequent states $x^1, x^2, ..., x^{T-1}$ along the time horizon are dependent on the inputs $u^1, u^2, ..., u^{T-1}$. The collection of states, $\mathbf{x}$, is the trajectory of the dynamical system, and is uniquely determined by x^0 and $\mathbf{u}$. This input-to-state map can be described using

$$G(x^0, \mathbf{u}) = \mathbf{x}, \tag{4}$$

where $G : \mathbb{R}^{n \times mT} \to \mathbb{R}^{nT}$ is a nonlinear and nonconvex reformulation of the system's dynamics. A similar input-to-state reformulation applied to inequality constraints can be written as $H(x^0, \mathbf{u}) \leq 0$, where $H : \mathbb{R}^{n \times mT} \to \mathbb{R}^p$, and are element-wise constraints. Here we use the fact that based on Equation (4), $\mathbf{x}$ can be expressed as a function of $\mathbf{u}$ alone. Note that G and H are defined implicitly, and that an explicit formula for them are typically challenging to determine and are not required for our proposed approach.

We re-express the objective in quadratic form as

$$\begin{aligned} & J(\mathbf{x}, \mathbf{u}) \\ & \quad = \frac{1}{2}\left(\begin{bmatrix}\mathbf{x}\\ \mathbf{u}\end{bmatrix} - \begin{bmatrix}\mathbf{x}_r\\ \mathbf{u}_r\end{bmatrix}\right)^\intercal \begin{bmatrix} Q & 0 \\ 0 & R \end{bmatrix} \left(\begin{bmatrix}\mathbf{x}\\ \mathbf{u}\end{bmatrix} - \begin{bmatrix}\mathbf{x}_r\\ \mathbf{u}_r\end{bmatrix}\right), \end{aligned} \tag{5}$$

where $Q \in \mathbb{R}^{nT \times nT}$ and $R \in \mathbb{R}^{mT \times mT}$ are symmetric positive definite weight matrices, and $\mathbf{x}_r \in \mathbb{R}^{nT}$ and $\mathbf{u}_r \in \mathbb{R}^{mT}$ are some constant vectors.

Combining the above reformulations, the trajectory optimization NLP takes on a new form

$$\begin{aligned} \min_{\mathbf{x}, \mathbf{u}} \quad & J(\mathbf{x}, \mathbf{u}) \\ \text{s.t.} \quad & G(x^0, \mathbf{u}) = \mathbf{x}, \\ & H(x^0, \mathbf{u}) \leq 0. \end{aligned} \tag{6}$$

To solve Equation (6), we propose a variant of SQP that linearizes the constraints using first order trajectory sensitivities, which makes it possible for us to accurately compute the derivative of the implicit input-to-state map G with respect to the input trajectory $\mathbf{u}$.

B. Trajectory-Sensitivities-based Linearization

Trajectory sensitivities arise naturally in the linearization of the input-to-state map G and H, using which the system dynamics can be linearized. Rearrange equation (4) to get

$$\mathbf{x} - G(x^0, \mathbf{u}) = 0,$$

and linearize at iteration s to get

$$\mathbf{x}_s - G(x^0, \mathbf{u}_s) + \mathbf{x}_{s+1} - \mathbf{x}_s - \left.\frac{dG}{d\mathbf{u}}\right|_s (\mathbf{u}_{s+1} - \mathbf{u}_s) = 0,$$

which can be simplified to

$$\mathbf{x}_{s+1} = \mathbf{x}_s + \left.\frac{dG}{d\mathbf{u}}\right|_s (\mathbf{u}_{s+1} - \mathbf{u}_s), \tag{7}$$

where $\left.\frac{dG}{d\mathbf{u}}\right|_s \in \mathbb{R}^{(nT \times mT) \times T}$ is the first-order trajectory sensitivities, which can be computed using Equation (2).

To perform trajectory-sensitivities-based linearization of H, first define

$$H(x^0, \mathbf{u}) = \mathbf{h},$$

where $\mathbf{h} \in \mathbb{R}^p$ is a vector whose entries correspond to values that H evaluate to. Then, inequality constraints can be linearized using trajectory sensitivities as

$$\mathbf{h}_s + \left.\frac{dH}{d\mathbf{u}}\right|_s (\mathbf{u}_{s+1} - \mathbf{u}_s) \leq 0. \tag{8}$$

Note that the linearization in Equation (7) and (8) is with respect to $\mathbf{u}$ alone, hence the value of any state x^k depends only on the initial state and inputs. However, in the multistage approximation, each state depends on the previous state, which allows numerical error to propagate through. Therefore, trajectory-sensitivities-based linearization has less approximation errors caused by numerical issue.

C. Trust Region Constraint

Linearization of constraints introduces approximation error. To effectively limit the magnitude of such error, a common method is to introduce a trust-region [6]. To do so, a suitable trust region radius Δ is selected to construct a constraint of the form

$$\|\mathbf{u}_{s+1} - \mathbf{u}_s\| \leq \Delta,$$

and define $d\mathbf{u} \in \mathbb{R}^{mT} := \mathbf{u}_{s+1} - \mathbf{u}_s$ for simplicity.

D. SQP-TS Algorithm

Combining the aforementioned modifications to the traditional SQP algorithm, a new variant of SQP with trajectory-sensitivities-based constraint linearization (SQP-TS) is proposed. This method solves a series of quadratic subproblems of the form

$$\begin{aligned} \min_{d\mathbf{u}} \quad & J(\mathbf{x}_{s+1}, \mathbf{u}_{s+1}) \\ \text{s.t.} \quad & \mathbf{x}_s + \left.\frac{dG}{d\mathbf{u}}\right|_s d\mathbf{u} = \mathbf{x}_{s+1}, \\ & \mathbf{h}_s + \left.\frac{dH}{d\mathbf{u}}\right|_s d\mathbf{u} \leq 0, \\ & \mathbf{u}_{s+1} = \mathbf{u}_s + d\mathbf{u}, \\ & \|d\mathbf{u}\| \leq \Delta, \end{aligned} \tag{9}$$

until $\|d\mathbf{u}\|$ approaches zero, and the algorithm converges. Note that $\mathbf{x}_s$, $\mathbf{u}_s$, and $\mathbf{h}_s$ are all constant vectors in the optimization problem at each iteration s, and the only decision variables are $d\mathbf{u}$.

The pseudocode of SQP-TS is presented in Algorithm 1. In the algorithm, we compute the exact system state trajectory $\mathbf{x}_s$ at line 3. Then, we compute the trajectory sensitivities at line 4. At line 5, we construct linear approximations of constraints. At line 6, we solve the quadratic subproblem. Finally, at lines 7 and 8, we update the new inputs $\mathbf{u}_{s+1}$. Convergence is assessed by comparing the change of a set of selected system variables to a predefined tolerance.

Algorithm 1 SQP-TS

1: Choose initial trajectory $(\mathbf{x}_0, \mathbf{u}_0)$; set $s \leftarrow 0$
2: **while** Convergence test not satisfied **do**
3: Recompute the state trajectory $\mathbf{x}_s$ by numerical iteration of the system dynamics in (1)
4: Compute the trajectory sensitivities by iteration of (2), then evaluate (3), at $\mathbf{u}_s$
5: Linearize the constraints using the trajectory sensitivities by (7) and (8)
6: Solve the quadratic program (9) to obtain $d\mathbf{u}$
7: Set $\mathbf{u}_{s+1} \leftarrow \mathbf{u}_s + d\mathbf{u}$ and $s \leftarrow s + 1$
8: **end while**

IV. Simulation

In this section we demonstrate the proposed approach in trajectory optimization of an autonomous vehicle. The proposed SQP-TS algorithm is implemented in Python. Each quadratic subproblem is solved using the commercially available Gurobi solver [13].

A. Application to a Vehicle Model

A vehicle is selected as the mobile agent of interest, and we consider the application of generating its optimal trajectory given some reference paths. To model the vehicle dynamics, a single-track bicycle model is used, which requires the following set of state variables and input:

$$x = \begin{bmatrix} v_x & v_y & \theta & \dot{\theta} & P_x & P_y \end{bmatrix}^\intercal, \quad u = \delta,$$

where $x \in \mathbb{R}^6$ is the state vector, $u \in \mathbb{R}$ is the input, v_x and v_y are longitudinal and lateral velocities of the vehicle in the global frame, θ and $\dot{\theta}$ are heading angle and angular velocity, P_x and P_y are longitudinal and lateral displacements in the global frame, and δ is the steering angle.

Following [14], some assumptions and simplifications are made to simplify the vehicle dynamic model, and the following continuous-time model is obtained

$$\dot{x} = \begin{bmatrix} -v_x\dot{\theta} + \frac{C_{\alpha f}(\frac{v_y + a\dot{\theta}}{v_x} - \delta)}{m} + \frac{C_{\alpha r}(\frac{v_y - b\dot{\theta}}{v_x})}{m} \\ v_y\dot{\theta} + \frac{s_f C_{xf} + s_r C_{xr}}{m} - \frac{C_{\alpha f}(\frac{\dot{y} - b\dot{\theta}}{v_x} - \delta)\delta}{m} \\ \dot{\theta} \\ \frac{1}{I_z}(aC_{\alpha f}(\frac{v_y - b\dot{\theta}}{v_x} - \delta) - bC_{\alpha r}(\frac{v_y - b\dot{\theta}}{v_x})) \\ v_x \sin\theta + v_y \cos\theta \\ v_x \cos\theta - v_y \sin\theta \end{bmatrix},$$

where all new variables are vehicle-related parameters, and their definitions can be found in [14]. Picking a fixed sample time T_s, this nonlinear model can be discretized into the form

$$x^{k+1} = g(x^k, u^k, T_s) \ \ \forall \ \ k \in \{0, ..., T-1\}. \tag{10}$$

Treating T_s as a constant, the expression above can then be reformulated as Equation (4).

The cost function is defined as

$$J_v(\mathbf{x},\mathbf{u}) = \frac{1}{2}w_1\sum_{k=0}^{T}\left\|\begin{bmatrix}P_x^k\\P_y^k\end{bmatrix}-\begin{bmatrix}P_{x,\text{ref}}^k\\P_{y,\text{ref}}^k\end{bmatrix}\right\|^2 + \frac{1}{2}w_3\sum_{k=0}^{T}(u^k)^2+\frac{1}{2}w_2\sum_{k=0}^{T-1}(u^{k+1}-u^k)^2,$$

where w_1, w_2, and w_3 are weight factors. The first term drives the vehicle along a reference trajectory, the second term limits control effort, and the third term reduces abrupt turning that may cause discomfort in passengers. Since this objective is quadratic, it can be rewritten into the form shown in Equation (5).

Observe that the trajectory optimization problem with a vehicle as mobile agent has a quadratic objective and nonlinear equality dynamic constraints. Such problem fits well with the problem formulation in (6), and can be solved effectively using the proposed method. Note that the infinity norm is used to construct the trust region constraints, because it has a linear implementation.

B. Initialization

Given no prior information on what inputs will solve the problem, some reasonable input and trajectory pair $(\mathbf{x_0},\mathbf{u_0})$ is selected to cold start the SQP iterations. By numerically integrating the system to obtain the state trajectory we can ensure the trajectory satisfies the equality constraints.

C. Extension to Collision Avoidance

With an extension to also consider obstacle avoidance, inequality constraints are needed. Recall that the initial input and trajectory pair $(\mathbf{x_0},\mathbf{u_0})$ is selected to cold start the SQP iterations, but it is possible that $(\mathbf{x_0},\mathbf{u_0})$ violates the inequality constraints. In that case, the trust-region condition will lead to infeasibility, as Δ may not be large enough to allow $\mathbf{u}$ to be modified such that $\mathbf{h}$ becomes non-positive.

To handle this potential infeasiblity, we propose to relax the inequality constraints to be

$$\mathbf{h}_s + \left.\frac{dH}{d\mathbf{u}}\right|_s (d\mathbf{u})_s \le \mathbf{r}_s, \tag{11}$$

where $\mathbf{r}_s \in \mathbb{R}^p$ is the constraint violation vector at the s-th iteration, whose entries $(r_i)_s \in \mathbb{R}$ are variables that measure the amount of constraint violation. For inequality constraints that cannot be relaxed, the corresponding r_i's are set to zero.

In the meantime, constraint violation must be reduced to zero before the algorithm converges. To achieve this, $\mathbf{r}_s$ is introduced into the objective function

$$J_r(\mathbf{r}_s) = \rho\mathbf{r}_s^\intercal\mathbf{r}_s,$$

where $J_r : \mathbb{R}^p \to \mathbb{R}$ evaluates cost of constraint violation and $\rho \in \mathbb{R}_{\ge 0}$ is the weight coefficient. This cost metric is quadratic and thus satisfies the assumption that the objective is quadratic. The weight ρ increases along with the number of SQP iterations, forcing inequality constraints to be prioritized in optimization and constraint violation to be eventually eliminated.

Finally, to implement collision avoidance constraints, we consider the case of static obstacles with known position and dimension. By modeling these obstacles as ellipses, the following constraints are constructed to prevent collision:

$$\frac{(P_x^k - C_x)^2}{A^2} + \frac{(P_y^k - C_y)^2}{B^2} \ge 1,$$

where C_x and C_y are coordinates of the center of ellipse modeling an obstacle, A and B are major and minor axes of the ellipse [15]. Such constraints are nonlinear and nonconvex, and will be reformulated into Equation (11). Here we choose ellipses as an example, while other less regular shapes with collision avoidance constraints that are differentiable with respect to the states are also acceptable.

V. Simulation Results

To demonstrate the performance of SQP-TS, we compare our method with the traditional trust-region SQP with first-order constraint linearization, which we denote by LSQP. We make this comparison to capture the impact of the trajectory sensitivities. In all subsequent test cases, unless otherwise defined, the initial vehicle position (P_x^0, P_y^0) is assumed to be equal to the initial position of reference, the initial vehicle heading θ^0 is selected to be $0\ rad$, and the initial trajectory is chosen to be the zero-input trajectory evaluated using (10). Vehicle parameters are fixed for all test cases, and are set to

$$C_{\alpha f} = C_{\alpha r} = -50000\ N/rad,\ C_{xf} = C_{xr} = 5000\ N,$$
$$s_f = s_r = 0.1,\ a = b = 1.2\ m,$$
$$m = 1200\ kg,\ I_z = 2000\ kg\cdot m^2.$$

A. Convergence Behavior and Trust-Region Radius

Fix start and end points, Dubins paths of variable turning radius are generated using the Python *Dubins* package. We select $(P_x^0, P_y^0, \theta^0) = (100, 100, 0.5)$ and $(P_x^T, P_y^T, \theta^T) = (105, 105, 2.5)$ where the unit is meters for coordinates and radians for angle. The minimum turning radius of the path is randomly sampled from the range of $(1, 6)$ radians with a step size of 0.3. Using these Dubins paths as references, both SQP-TS and LSQP are applied to solve the vehicle trajectory optimization problem. The relevant parameters used in solving this test case are

$$T = 80\ s,\ T_s = 0.01\ s,\ \tau = 1.0,$$
$$w_1 = 100,\ w_2 = 20,\ w_3 = 0$$

where τ is the tolerance of change in vehicle's paths of two subsequent iterations, and the algorithm terminates when τ is not exceeded. Another important parameter is the trust region radius, Δ, whose magnitude will be actively adjusted based on the performance of both algorithms.

A total of 60 Dubins paths are generated as reference paths. As SQP-TS and LSQP are used to perform trajectory optimization over these references, it is noticed that the maximum allowable Δ without divergence is 0.5 for SQP-TS, and 0.3 for LSQP. Therefore, we first set $\Delta = 0.3$ to run both algorithms, and then set $\Delta = 0.5$ to run SQP-TS. To make a comparison, we record the total iterations to converge

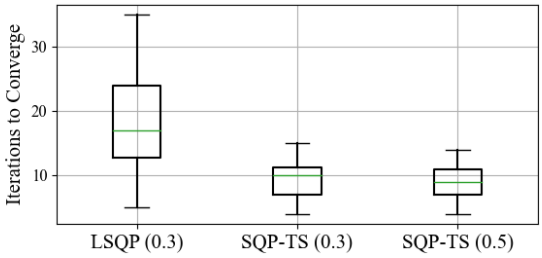

Fig. 1. Total iterations to converge. LSQP is shown for a trust region of 0.3, while SQP-TS is shown for a trust region of both 0.3 and 0.5.

and collect them into the box plot in Figure 1. The numbers in the x labels represent the magnitude of Δ. When both LSQP and SQP-TS are executed at $\Delta = 0.3$, SQP-TS takes significantly less number of iterations to converge than LSQP. When SQP-TS is ran at $\Delta = 0.5$, its iterations to converge is slightly reduced, in comparison to the same algorithm ran at $\Delta = 0.3$. The fact that SQP-TS is allowed a larger trust-region radius without divergence is contributed by the trajectory-sensitivities-based linearization of constraints of Problem (6) and its improvement in accuracy of approximation. With increased trust-region radius, $\mathbf{u}$ is able to converge in less number of iterations. Time per iteration is discussed in the next section.

B. Convergence Behavior and Initial Inputs

Using 20 of the Dubins paths generated in the previous test case, we want to explore how initial inputs affect the performance of both methods. The initial trajectories are computed by numerical integration of Equation (10) with the following choice of constant inputs

$$u^k = 0, 0.1, 0.2, 0.3\ rad/s^2,\ \ \forall\ \ k \in \{0, ..., T-1\}.$$

Setting $\Delta = 0.3$, $\tau = 1$, and executing both SQP-TS and LSQP, the iterations to converge and total solver time to converge are recorded. Solver time is the time used to solve each optimization subproblem reported by Gurobi, and does not account for the time spend building the problem model. Note that both SQP-TS and LSQP spends 0.03 s per iteration to perform constraint linearization, while SQP-TS computes trajectory sensitivities and LSQP computes gradients. Therefore, both methods spend similar amount of time constructing the subproblem. The results are shown in Figure 2.

As initial inputs change, the iterations to converge and total solver time of SQP-TS is relatively constant, while those of LSQP change drastically. For the extreme case of $u^k = 0.3$ for each k, LSQP takes about 11 times of iterations to converge than SQP-TS, and about 6 times of solver time. Therefore, SQP-TS is not as sensitive to initial inputs as LSQP is, and has consistent performance. Moreover, the fact that SQP-TS works with zero-input initialization demonstrates its good convergence behaviour. To further improve convergence speed, the algorithm can be warm started with trajectories computed using (10) with $\mathbf{u}$ generated by other

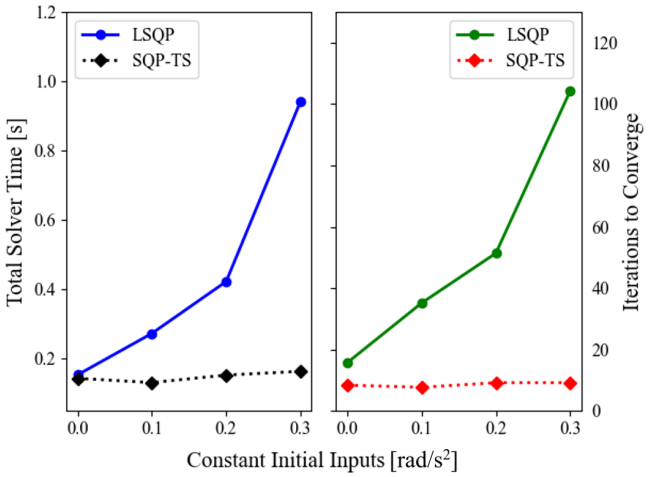

Fig. 2. Total solver time (blue and black lines) and iterations to converge (green and red lines) for different constant initial inputs using LSQP (solid lines) and SQP-TS (dotted lines) methods.

controllers, such as a linear model predictive controller (LMPC) [14] .

C. Example with Collision Avoidance

Now we extend the trajectory optimization problem to include static obstacles. To start, a simple scenario where the reference path is a straight line passing through one obstacle is considered. This test scenario is adapted from the example in [15]. The following parameters are used

$$\Delta = 0.3\ rad/s^2,\ \tau = 0.3,\ \rho = (1000)(s),$$
$$T = 80\ s,\ T_s = 0.02\ s,\ w_1 = 1000,\ w_2 = 1,\ w_3 = 0$$

where the weight coefficient of collision cost ρ is a variable proportional to the number of past algorithm iterations s. For this example, we set the initial vehicle heading to align with the reference path. Using LSQP and SQP-TS to solve this problem, the final vehicle paths are plotted in Figure 3.

A second reference path is selected to be a Dubin's curve generated using the same approach as does in the previous section, which intersects a static obstacle. The same set of parameters are used except for $T_s = 0.05\ s$, and LSQP and SQP-TS are applied to solve this problem. The final vehicle paths are plotted in Figure 3, where we see that both algorithms converge to the same final paths. For the straight reference path, LSQP takes 80 iterations to converge, while SQP-TS takes 12 iterations and converges faster. For the Dubin's reference, both methods take 8 iterations to converge.

Finally, consider a complex, nonsmooth reference path composed of straight lines, curves, sharp turns, and which passes through obstacles. The zero-input initialization is an infeasible path that crosses obstacle(s). The same set of parameters as the straight-line test scenario is used, with the following changes

$$\Delta = 0.02\ rad^2/s,\ \tau = 4,\ T = 200\ s,\ T_s = 0.1\ s.$$

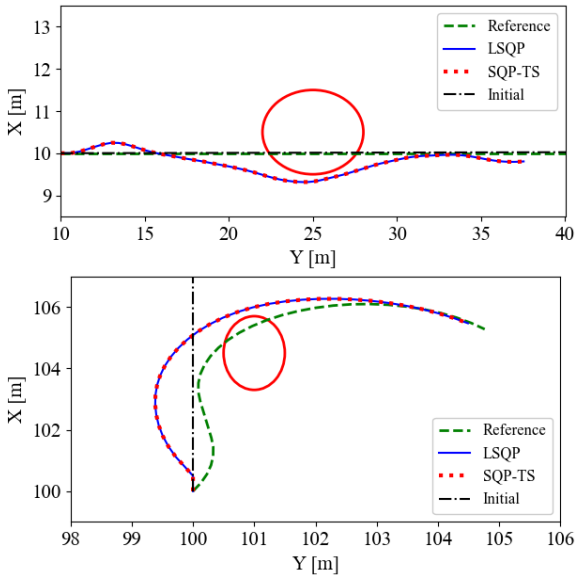

Fig. 3. Straight and Dubin's reference path with an obstacle in which both the proposed method and LSQP converge.

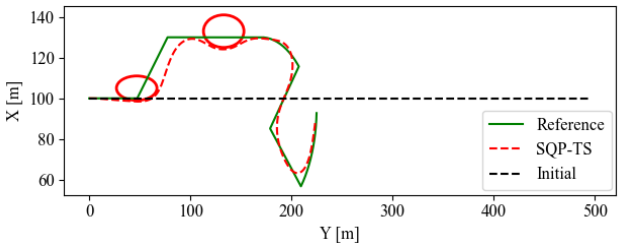

Fig. 4. Complex path with multiple obstacles in which SQP-TS converges but LSQP fails.

A larger τ is used because the length of the path is longer than the straight line or the Dubin's path. A smaller trust-region radius is selected because the reference path is more complex, and more accurate approximation is needed. For this test scenario, LSQP fails due to numerical error at 100 iterations. SQP-TS converges in 14 iterations, and the final vehicle path is shown in Figure 4.

Closer investigation reveals that the numerical error is likely caused by approximation error in the states impacting the collision avoidance constraints. In LSQP, this eventually leads to constraint violation and causes numerical trouble. Multiple different obstacle arrangements have been tested, but LSQP always fails due to the same issue. On the other hand, SQP-TS solves the optimization problem with more accurate approximation, and no such failure has been observed. Therefore, SQP-TS is better at handling collision avoidance constraints with a complex reference path than the LSQP method, and is not as prone to convergence failure.

VI. Convergence Analysis

Here we present a convergence analysis of the proposed SQP-TS algorithm. To begin, consider a local minimum of the NLP problem (6) at $(\mathbf{x}^*, \mathbf{u}^*)$. Let $\bar{H}$ be the subset of H consisting of the active constraints at $(\mathbf{x}^*, \mathbf{u}^*)$, and define $\bar{G}(\mathbf{x}, \mathbf{u}) = \begin{bmatrix}(\mathbf{x} - G(x^0, \mathbf{u}))^\intercal & \bar{H}(x^0, \mathbf{u})^\intercal\end{bmatrix}^\intercal$ consisting of $\bar{n}$ constraints.

Assumption 1. $d\bar{G}$ is full rank at $(\mathbf{x}^*, \mathbf{u}^*)$. Also known as Linear Independence Constraint Qualification (LICQ), this holds almost always by Sard's Theorem [6] [16].

We define $\mathbf{q} = \begin{bmatrix}\mathbf{x}^\intercal & \mathbf{u}^\intercal\end{bmatrix}^\intercal$, $\mathbf{q}_r = \begin{bmatrix}\mathbf{x}_r^\intercal & \mathbf{u}_r^\intercal\end{bmatrix}^\intercal$, and P a block diagonal matrix defined as $P = diag(Q, R)$. By removing the inactive inequality constraints and considering the active inequality constraints as equality constraints, the original problem in (6) is equivalent to

$$\begin{aligned} \min_{\mathbf{q}} \quad & \frac{1}{2}(\mathbf{q} - \mathbf{q}_r)^\intercal P(\mathbf{q} - \mathbf{q}_r) \\ \text{s.t.} \quad & \bar{G}(\mathbf{q}) = 0. \end{aligned} \tag{12}$$

By removing the trust region constraints in (9), we get

$$\begin{aligned} \min_{d\mathbf{u}} \quad & J\left(\mathbf{x}_{s+1}, \mathbf{u}_{s+1}\right) \\ \text{s.t.} \quad & \mathbf{x}_s + \left.\frac{dG}{d\mathbf{u}}\right|_s d\mathbf{u} = \mathbf{x}_{s+1}, \\ & \mathbf{h}_s + \left.\frac{dH}{d\mathbf{u}}\right|_s d\mathbf{u}_s \leq 0, \\ & \mathbf{u}_{s+1} = \mathbf{u}_s + d\mathbf{u}_s. \end{aligned} \tag{13}$$

The following result from [17] shows that the same constraints are active locally when making our approximation.

Lemma 1. *Under the assumptions of LICQ, strict complementarity condition, and second-order sufficient conditions, if* $\mathbf{q}$ *is sufficiently close to* $\mathbf{q}^* = (\mathbf{x}^*, \mathbf{u}^*)$*, there is a solution of the subproblem (13) whose set of active inequality constraints is the same as that for (6) at* $(\mathbf{x}^*, \mathbf{u}^*)$ *[17].*

By Lemma 1, there exists $r > 0$ such that $\|\mathbf{q} - \mathbf{q}^*\| < r$ implies that the active constraints in (13) are the linearization of $\bar{G}$ at $(\mathbf{x}^*, \mathbf{u}^*)$. Thus, (13) is equivalent to the following equality-constrained problem

$$\begin{aligned} \min_{\hat{\mathbf{q}}} \quad & \frac{1}{2}(\hat{\mathbf{q}} - \mathbf{q}_r)^\intercal P(\hat{\mathbf{q}} - \mathbf{q}_r) \\ \text{s.t.} \quad & \bar{G}(\mathbf{q}) + d\bar{G}_q^\intercal(\hat{\mathbf{q}} - \mathbf{q}) = 0. \end{aligned} \tag{14}$$

As $\bar{G}$ is C^2, the function $\begin{bmatrix}\bar{G}^\intercal & I\end{bmatrix}^\intercal$ is Lipschitz with some Lipschitz constant $L > 0$. The following theorem shows that the solutions to (14) are contracting towards $\mathbf{q}^*$.

Theorem 2. *Under Assumption 1, there exists* $\hat{r} > 0$ *such that* $\|\mathbf{q}_s - \mathbf{q}^*\| < \hat{r}$ *implies that the solution to (14) satisfies:*

$$\|\hat{\mathbf{q}} - \mathbf{q}^*\| \leq k \|\mathbf{q} - \mathbf{q}^*\|, \; k, kL \in (0, 1). \tag{15}$$

Proof Sketch. For Problem (12), introduce Lagrange multipliers $\lambda \in \mathbb{R}^{nT}$ and write the Lagrangian, take its derivative, set to zero to get

$$P(\mathbf{q}^* - \mathbf{q}_r) = -d\bar{G}_{\mathbf{q}^*}\lambda^*, \tag{16}$$

where λ^* are the Lagrange multipliers at $\mathbf{q}^*$. Let $w_q = d\bar{G}_{\mathbf{q}}^\intercal$.

For Problem (14), introduce Lagrange multipliers $\hat{\lambda} \in \mathbb{R}^{nT}$, write the Lagrangian, take its derivative and set to zero to get

$$\begin{bmatrix} P & w_q \\ w_q^\intercal & 0 \end{bmatrix} \begin{bmatrix} \hat{\mathbf{q}} \\ \hat{\lambda} \end{bmatrix} = \begin{bmatrix} P\mathbf{q}_r \\ w_q^\intercal \mathbf{q} - \bar{G}(\mathbf{q}) \end{bmatrix}.$$

Inverting the matrix on the left hand side, the solution to this system can be shown to be

$$\begin{aligned}\hat{\mathbf{q}} \;=\; & P^{-1}\left(I - w_q(w_q^\intercal P^{-1} w_q)^{-1} w_q^\intercal P^{-1}\right) P\mathbf{q}_r \\ & + P^{-1} w_q \left(w_q^\intercal P^{-1} w_q\right)^{-1} \left(w_q^\intercal \mathbf{q} - \bar{G}(\mathbf{q})\right) =: f(\mathbf{q}).\end{aligned} \tag{17}$$

By Assumption 1, there exists $\hat{r} > 0$ such that for all $\mathbf{q}$ such that $\|\mathbf{q} - \mathbf{q}^*\| < \hat{r}$, $w_q \neq 0$. As P is symmetric positive definite, this implies that Equation (17) is well-defined so there exists a unique solution $\hat{\mathbf{q}}$.

By the multivariate mean value theorem, for each i there exists $\mathbf{q}^i = t_i\mathbf{q} + (1-t_i)\mathbf{q}^*$ for some $t_i \in [0,1]$ such that $\bar{G}_i(\mathbf{q}) = \frac{d\bar{G}_i}{d\mathbf{q}}\Big|_{\mathbf{q}^i}(\mathbf{q} - \mathbf{q}^*)$. Define $\bar{w}_q^\intercal := \begin{bmatrix} \frac{d\bar{G}_1}{d\mathbf{q}}\Big|_{\mathbf{q}^1}^\intercal & \dots & \frac{d\bar{G}_{\bar{n}}}{d\mathbf{q}}\Big|_{\mathbf{q}^{\bar{n}}}^\intercal \end{bmatrix}^\intercal$, then we have

$$\bar{G}(\mathbf{q}) = \bar{w}_q^\intercal(\mathbf{q} - \mathbf{q}^*). \tag{18}$$

Subtracting $\mathbf{q}^*$ from $\hat{\mathbf{q}}$ and substituting in Equation (16) and (18), it can be shown that

$$\begin{aligned}\hat{\mathbf{q}} - \mathbf{q}^* = & P^{-1} w_q \left(w_q^\intercal P^{-1} w_q\right)^{-1} (w_q - \bar{w}_q)^\intercal(\mathbf{q} - \mathbf{q}^*) \\ & + P^{-1} w_q \left(w_q^\intercal P^{-1} w_q\right)^{-1} w_q^\intercal P^{-1}(w_{q^*} - w_q)(-\lambda^*) \\ & + P^{-1}(w_q - w_{q^*})(-\lambda^*).\end{aligned} \tag{19}$$

As $\left\|d\bar{G}_q\right\|$ is continuous over $\overline{B}_{\hat{r}}(\mathbf{q}^*)$ compact, by shrinking r if necessary, there exist $c_m > 0$ and $c_M > 0$ such that

$$0 < c_m \leq \left\|d\bar{G}_q\right\| \leq c_M \tag{20}$$

for all $\mathbf{q} \in \overline{B}_{\hat{r}}(\mathbf{q}^*)$. Let $\lambda_{\max}$ and $\lambda_{\min}$ denote the maximum and minimum eigenvalues of P, shrink r if necessary and choose $d > 0$ such that

$$k := \frac{\lambda_{\max}^2 c_M}{\lambda_{\min}^2 c_m^2}(d + 2rL) < \min\{L^{-1}, 1\}. \tag{21}$$

As $\left\|d\bar{G}_q\right\|$ is continuous over $\overline{B}_r(\mathbf{q}^*)$ compact, it is uniformly continuous, so there exists $\alpha > 0$ such that $\|\mathbf{q} - \mathbf{p}\| < \alpha$ implies that

$$\left\|d\bar{G}_q - d\bar{G}_p\right\| < d. \tag{22}$$

Now expressing $\hat{\mathbf{q}} - \mathbf{q}^*$ as Equation (19), and bounding the norm of each term on the right hand side using (20), (21), and (22), it can be shown that

$$\|\hat{\mathbf{q}} - \mathbf{q}^*\| \leq k \|\mathbf{q} - \mathbf{q}^*\|.$$

Next, write $\hat{\mathbf{q}} = \begin{bmatrix}\hat{\mathbf{x}}^\intercal & \hat{\mathbf{u}}^\intercal\end{bmatrix}^\intercal$, we will show that the contraction of $\hat{\mathbf{q}}$ towards $\mathbf{q}^*$ obtained from Theorem 2 also leads to a contraction of $\hat{\mathbf{u}}$ towards $\mathbf{u}^*$ alone.

Corollary 3. *Under the conditions of Theorem 2, there exists $\hat{r} > 0$ such that $\|\mathbf{u} - \mathbf{u}^*\| < \hat{r}$ and $\mathbf{x} = G(x^0, \mathbf{u})$ implies that the solution to (14) satisfies:*

$$\|\hat{\mathbf{u}} - \mathbf{u}^*\| \leq k \|\mathbf{u} - \mathbf{u}^*\|, \; k \in (0,1). \tag{23}$$

Proof. Define $F := \begin{bmatrix}\bar{G}^\intercal & I\end{bmatrix}^\intercal$ and recall F is Lipschitz with Lipschitz constant L.

Applying Theorem 2, we compute

$$\begin{aligned}\|\hat{\mathbf{u}} - \mathbf{u}^*\| &\leq \left\|\begin{bmatrix}\hat{\mathbf{x}} - \mathbf{x}^* \\ \hat{\mathbf{u}} - \mathbf{u}^*\end{bmatrix}\right\| \leq \bar{k}\left\|\begin{bmatrix}\mathbf{x} - \mathbf{x}^* \\ \mathbf{u} - \mathbf{u}^*\end{bmatrix}\right\| \\ &= \bar{k}\left\|\begin{bmatrix}G(x^0, \mathbf{u}) - G(x^0, \mathbf{u}^*) \\ \mathbf{u} - \mathbf{u}^*\end{bmatrix}\right\| \\ &= \bar{k}\left\|F(x^0, \mathbf{u}) - F(x^0, \mathbf{u}^*)\right\| \\ &\leq \bar{k}L\left\|\mathbf{u} - \mathbf{u}^*\right\|,\end{aligned}$$

where $k := \bar{k}L < 1$ by Theorem 2. □

Recall that problem (9) is the subproblem (13) with trust region constraints included. By Lemma 1, for $\mathbf{q}_s \in B_r(\mathbf{q}^*)$, problem (9) is equivalent to

$$\begin{aligned}\min_{\mathbf{q}_{s+1}} \quad & \frac{1}{2}(\mathbf{q}_{s+1} - \mathbf{q}_r)^\intercal P(\mathbf{q}_{s+1} - \mathbf{q}_r) \\ \text{s.t.} \quad & \bar{G}(\mathbf{q}_s) + d\bar{G}_{q_s}^\intercal(\mathbf{q}_{s+1} - \mathbf{q}_s) = 0, \\ & \|\mathbf{u}_{s+1} - \mathbf{u}_s\| \leq \Delta.\end{aligned} \tag{24}$$

The following theorem provides a local convergence guarantee for Algorithm 1.

Theorem 4. *Suppose $H(x^0, \mathbf{u}) = M\mathbf{u}$ for some constant matrix M. Assume the conditions of Lemma 1 and Corollary 3. There exist $\bar{r} > \hat{r}$ and $\bar{\Delta} > 0$ such that for any $\Delta \in (0, \bar{\Delta})$ and any $\mathbf{u}_0$ such that $\|\mathbf{u}_0 - \mathbf{u}^*\| \leq \bar{r}$, the sequence of solutions obtained from subproblem (9), denoted $\{\mathbf{u}_s\}_{s=0}^\infty$, converges to $\mathbf{u}^*$.*

Remark 5. This theorem provides a local convergence guarantee for the proposed method in the important special case where H is a linear function of inputs, as in the example provided in Section V.A and V.B. Future work will aim to extend this to more general inequality constraints H.

Proof. By Equation (17), $\hat{\mathbf{u}} = f(\mathbf{u})$ is continuous. By Corollary 3, f is a contraction over $B_{\hat{r}}(\mathbf{u}^*)$. Thus, there exists $\tilde{r} > \hat{r}$ such that $\mathbf{u} \in B_{\tilde{r}}(\mathbf{u}^*)$ implies that $f(\mathbf{u}) \in \overline{B}_{\hat{r}}(\mathbf{u}^*)$. Again, since $f(\mathbf{u})$ is continuous, $\|\hat{\mathbf{u}} - \mathbf{u}\| = \|f(\mathbf{u}) - \mathbf{u}\|$ achieves a minimum over the compact set $\overline{B}_{\tilde{r}}(\mathbf{u}^*) - B_{\hat{r}}(\mathbf{u}^*)$, say c. By Corollary 3, f is a contraction over $B_{\hat{r}}(\mathbf{u}^*)$ with a unique fixed point $f(\mathbf{u}^*) = \mathbf{u}^*$, so $c > 0$ since f does not have a fixed point on the set $\overline{B}_{\tilde{r}}(\mathbf{u}^*) - B_{\hat{r}}(\mathbf{u}^*)$. Let $\bar{\Delta} = c$. Fix any $\Delta \in (0, \bar{\Delta})$.

Since H is linear, and any solution $\mathbf{u}_s$ to Problem (9) satisfies (8), $H(x^0, \mathbf{u}_s) \leq 0$. By Algorithm 1 step 3, $\mathbf{x}_s = G(x^0, \mathbf{u}_s)$. Therefore, $\bar{G}(\mathbf{q}_s) = \bar{G}(\mathbf{x}_s, \mathbf{u}_s) = 0$. Recall that Problem (9) is equivalent to Problem (24) based on our assumptions, which implies that any solution $\mathbf{q}_s$ to Problem (24) satisfies $\bar{G}(\mathbf{q}_s) = 0$.

Recall that without trust region constraints, $\hat{\mathbf{q}}$ is the solution to subproblem (14). Measuring distance using the inner product induced by P, it follows that the solution $\hat{\mathbf{q}}$ to Problem (14) with $\mathbf{q} = \mathbf{q}_s$ is the unique closest point on the hyperplane of $\bar{G}(\mathbf{q}_s) + d\bar{G}_{q_s}^\intercal(\mathbf{q} - \mathbf{q}_s) = d\bar{G}_{q_s}^\intercal(\mathbf{q} - \mathbf{q}_s) = 0$ to $\mathbf{q}_r$. Let S denote the intersection of $d\bar{G}_{q_s}^\intercal(\mathbf{q} - \mathbf{q}_s) = 0$ and $\mathbb{R}^{nT} \times \overline{B}_\Delta(\mathbf{u}_s)$. The solution $\mathbf{q}_{s+1}$ to (24) is the unique closest point in S to $\mathbf{q}_r$. Since $\mathbf{q}_s \in S$ because $\bar{G}(\mathbf{q}_s) = 0$, it is straight forward to show that the closest point in S to

$\mathbf{q}_r$ is also the closest point in S to $\hat{\mathbf{q}}$. This latter point is equal to the intersection of the line segment from $\mathbf{q}_s$ to $\hat{\mathbf{q}}$ and the boundary of $\mathbb{R}^{nT} \times \overline{B}_\Delta(\mathbf{u}_s)$. A point on this line segment is

$$(1-\gamma)\mathbf{q}_s + \gamma\hat{\mathbf{q}} = \mathbf{q}(\gamma) = \begin{bmatrix}\mathbf{x}(\gamma)^\intercal & \mathbf{u}(\gamma)^\intercal\end{bmatrix}^\intercal, \tag{25}$$

where $\gamma \in (0,1)$. Then $\mathbf{q}(\gamma)$ intersects the boundary of the $\mathbb{R}^{nT} \times \overline{B}_\Delta(\mathbf{q}_s)$ if and only if $\|\mathbf{u}(\gamma) - \mathbf{u}_s\| = \Delta$. Substituting in (25) and solving for γ yields $\hat{\gamma} = \frac{\Delta}{\|\hat{\mathbf{u}} - \mathbf{u}_s\|}$, so $\mathbf{u}(\hat{\gamma})$ becomes

$$\mathbf{u}(\hat{\gamma}) = (1-\hat{\gamma})\mathbf{u}_s + \hat{\gamma}\hat{\mathbf{u}}, \tag{26}$$

then

$$\mathbf{u}_{s+1} = \begin{cases} \hat{\mathbf{u}} & \text{if } \|\hat{\mathbf{u}} - \mathbf{u}_s\| \le \Delta \\ \mathbf{u}(\hat{\gamma}) & \text{if } \|\hat{\mathbf{u}} - \mathbf{u}_s\| > \Delta. \end{cases} \tag{27}$$

When $\mathbf{u}_{s+1} = \mathbf{u}(\hat{\gamma})$, subtract $\mathbf{u}^*$ from $\mathbf{u}_{s+1}$ to get

$$\mathbf{u}_{s+1} - \mathbf{u}^* = (1-\hat{\gamma})(\mathbf{u}_s - \mathbf{u}^*) + \hat{\gamma}(\hat{\mathbf{u}} - \mathbf{u}^*).$$

Since $\hat{\mathbf{u}} \in B_{\hat{r}}(\mathbf{u}^*)$, taking the norm and substitute Equation (23) in, we get

$$\|\mathbf{u}_{s+1} - \mathbf{u}^*\| \le (1 - \hat{\gamma} + k\hat{\gamma}) \|\mathbf{u}_s - \mathbf{u}^*\|.$$

As $\hat{\gamma}$ is continuous, and the set $\overline{B}_{\bar{r}}(\mathbf{u}^*) - B_{\hat{r}}(\mathbf{u}^*)$ is compact, $\hat{\gamma}$ attains a minimum $\bar{\gamma}$ over the set. Then $\bar{\gamma} < 1$ by the choice of Δ, which implies that $1 - \hat{\gamma} + k\hat{\gamma} \le 1 - \bar{\gamma} + k\bar{\gamma} < 1$, so we have a contraction for $\mathbf{u} \in \overline{B}_{\bar{r}}(\mathbf{u}^*) - B_{\hat{r}}(\mathbf{u}^*)$. Thus, if $\mathbf{u}_0$ is initially in the set $B_{\bar{r}}(\mathbf{u}^*)$, $\mathbf{u}_s$ will enter $B_{\hat{r}}(\mathbf{u}^*)$ in a finite number of iterations.

For any $\mathbf{u}_s \in B_{\hat{r}}(\mathbf{u}^*)$, by (27), $\mathbf{u}_{s+1}$ is equal to either $\hat{\mathbf{u}}$ or $\mathbf{u}(\hat{\gamma})$, so $\|\mathbf{u}_{s+1} - \mathbf{u}^*\| \le \hat{k} \|\mathbf{u}_s - \mathbf{u}^*\|$ where $\hat{k}$ is either k or $1 - \hat{\gamma} + k\hat{\gamma}$ and is less than 1. Thus, $\|\mathbf{u}_s - \mathbf{u}^*\|$ is monotonically decreasing in s and bounded below by 0, so $\|\mathbf{u}_s - \mathbf{u}^*\|$ converges, say to $\hat{c}$. Assume towards a contradiction that $\hat{c} > 0$. Then there exists $\tilde{\mathbf{u}}$ such that $\|\tilde{\mathbf{u}} - \mathbf{u}^*\| = \hat{c}$ and a subsequence of $\{\mathbf{u}_s\}_{s=0}^\infty$ converges to $\tilde{\mathbf{u}}$. Let $\tilde{\mathbf{u}}_+$ be the solution of (24) with $\tilde{\mathbf{u}}$ in place of $\mathbf{u}_s$. Then by the same argument as above, $\|\tilde{\mathbf{u}}_+ - \mathbf{u}^*\| \le \hat{k} \|\tilde{\mathbf{u}} - \mathbf{u}^*\|$ where $\hat{k} < 1$. As the solution of (24), which is given by (27), is piecewise continuous, and since a subsequence of $\{\mathbf{u}_s\}_{s=0}^\infty$ converges to $\tilde{\mathbf{u}}$, there exists $\mathbf{u}_s$ such that $\|\mathbf{u}_s - \tilde{\mathbf{u}}\| < \epsilon_1$ and that $\|\mathbf{u}_{s+1} - \mathbf{u}^*\| \le (\hat{k} + \epsilon_2) \|\mathbf{u}_s - \mathbf{u}^*\|$ for some $\epsilon_1, \epsilon_2 > 0$ such that $\hat{k} + \epsilon_2 < 1$ and $(\hat{k} + \epsilon_2)(\hat{c} + \epsilon_1) < \hat{c}$. Thus, $\|\mathbf{u}_s - \mathbf{u}^*\| < \hat{c} + \epsilon_1$, so $\|\mathbf{u}_{s+1} - \mathbf{u}^*\| < (\hat{k} + \epsilon_2)(\hat{c} + \epsilon_1) < \hat{c}$, which is a contradiction. So, $\hat{c} = 0$, which implies that $\{\mathbf{u}_s\}_{s=0}^\infty$ converges to $\mathbf{u}^*$.

□

VII. Conclusion

In this work, we propose an input-to-state reformation and the SQP-TS algorithm to solve trajectory optimization problems, and apply this method to generate paths for an autonomous vehicle. This method shows promises as it possesses both local theoretical guarantees and requires significantly fewer iterations to converge than the traditional SQP solvers, thanks to the improved quality of local approximation by using trajectory sensitivities to reduce numerical errors in the states.

As a next step, we aim to extend the proof to a broader case where the inequality constraints are nonlinear in the inputs. Also, we would like to explore the proposed algorithm's compatibility with a variety of nonlinear systems, with different formulation of obstacle avoidance constraints, and with other cost metrics. Reference paths generated using existing path planners may be used to further investigate the algorithm's path smoothing capability.

Finally, while the solution discussed in this paper is based on an SQP formulation, trajectory sensitivities and their corresponding local approximation with higher accuracy could also be applied to quasi-Newton or other trajectory optimization algorithms. We hope to explore the benefits of using trajectory sensitivities to improve numerical performance of these other methods as well.

References

[1] D. Malyuta, Y. Yu, P. Elango, and B. Açıkmeşe, "Advances in trajectory optimization for space vehicle control," *Annual Reviews in Control*, vol. 52, pp. 282–315, 2021.

[2] R. Chai, A. Savvaris, A. Tsourdos, S. Chai, R. Chai, A. Savvaris, A. Tsourdos, and S. Chai, "Overview of trajectory optimization techniques," *Design of Trajectory Optimization Approach for Space Maneuver Vehicle Skip Entry Problems*, pp. 7–25, 2020.

[3] Z. Wang, X. Zhou, C. Xu, and F. Gao, "Geometrically constrained trajectory optimization for multicopters," *IEEE Transactions on Robotics*, vol. 38, no. 5, pp. 3259–3278, 2022.

[4] T. A. Howell, S. Le Cleac'h, S. Singh, P. Florence, Z. Manchester, and V. Sindhwani, "Trajectory optimization with optimization-based dynamics," *IEEE Robotics and Automation Letters*, vol. 7, no. 3, pp. 6750–6757, 2022.

[5] A. V. Rao, "Trajectory optimization: a survey," *Optimization and Optimal Control in Automotive Systems*, pp. 3–21, 2014.

[6] J. Nocedal and S. J. Wright, *Numerical Optimization*. Springer, 1999.

[7] A. Wächter and L. T. Biegler, "On the implementation of an interior-point filter line-search algorithm for large-scale nonlinear programming," *Mathematical programming*, vol. 106, pp. 25–57, 2006.

[8] A. Zanelli, A. Domahidi, J. Jerez, and M. Morari, "Forces nlp: An efficient implementation of interior-point methods for multistage nonlinear nonconvex programs," *International Journal of Control*, vol. 93, no. 1, pp. 13–29, 2020.

[9] R. Chai, A. Savvaris, A. Tsourdos, S. Chai, and Y. Xia, "A review of optimization techniques in spacecraft flight trajectory design," *Progress in aerospace sciences*, vol. 109, p. 100543, 2019.

[10] J. Schulman, Y. Duan, J. Ho, A. Lee, I. Awwal, H. Bradlow, J. Pan, S. Patil, K. Goldberg, and P. Abbeel, "Motion planning with sequential convex optimization and convex collision checking," *The International Journal of Robotics Research*, vol. 33, no. 9, pp. 1251–1270, 2014.

[11] I. A. Hiskens and M. A. Pai, "Trajectory sensitivity analysis of hybrid systems," *IEEE Transactions on Circuits and Systems I: Fundamental Theory and Applications*, vol. 47, no. 2, pp. 204–220, 2000.

[12] M. W. Fisher and I. A. Hiskens, "Numerical computation of critical system recovery parameter values by trajectory sensitivity maximization," in *IEEE Conference on Decision and Control (CDC)*, 2019, pp. 8000–8006.

[13] Gurobi Optimization, LLC, "Gurobi Optimizer Reference Manual," 2024. [Online]. Available: https://www.gurobi.com

[14] S. Chen, H. Chen, and D. Negrut, "Implementation of mpc-based path tracking for autonomous vehicles considering three vehicle dynamics models with different fidelities," *Automotive Innovation*, vol. 3, no. 4, pp. 386–399, 2020.

[15] U. Rosolia, S. De Bruyne, and A. G. Alleyne, "Autonomous vehicle control: A nonconvex approach for obstacle avoidance," *IEEE Transactions on Control Systems Technology*, vol. 25, no. 2, pp. 469–484, 2016.

[16] J. M. Lee and J. M. Lee, *Smooth manifolds*. Springer, 2012.

[17] S. M. Robinson, "Perturbed kuhn-tucker points and rates of convergence for a class of nonlinear-programming algorithms," *Mathematical Programming*, vol. 7, pp. 1–16, 1974.