Bi-Convex Approximation of Non-Holonomic Trajectory Optimization

Arun Kumar Singh [1], Raghu Ram Theerthala[2], Mithun Babu[2], Unni Krishnan R Nair[2], K.Madhava Krishna[2]

***Abstract*— Autonomous cars and fixed-wing aerial vehicles have the so-called non-holonomic kinematics which non-linearly maps control input to states. As a result, trajectory optimization with such a motion model becomes highly non-linear and non-convex. In this paper, we improve the computational tractability of non-holonomic trajectory optimization by reformulating it in terms of a set of bi-convex cost and constraint functions along with a non-linear penalty. The bi-convex part acts as a relaxation for the non-holonomic trajectory optimization while the residual of the penalty dictates how well its output obeys the non-holonomic behavior. We adopt an alternating minimization approach for solving the reformulated problem and show that it naturally leads to the replacement of the challenging non-linear penalty with a globally valid convex surrogate. Along with the common cost functions modeling goal-reaching, trajectory smoothness, etc., the proposed optimizer can also accommodate a class of non-linear costs for modeling goal-sets, while retaining the bi-convex structure. We benchmark the proposed optimizer against off-the-shelf solvers implementing sequential quadratic programming and interior-point methods and show that it produces solutions with similar or better cost as the former while significantly outperforming the latter. Furthermore, as compared to both off-the-shelf solvers, the proposed optimizer achieves more than 20x reduction in computation time.**

I. Introduction

The computational ease with which a trajectory optimization can be solved depends critically on the motion model of the robot. For example, for quadrotors which can be modeled as an affine system (series of integrators), generating a smooth trajectory through a set of waypoints takes the shape of a convex quadratic programme [1]. Even in the presence of obstacles, trajectory optimization can be shown to have the difference of convex form which can be solved efficiently using the convex-concave procedure [2], [3]. In contrast, autonomous cars and fixed wing aerial vehicles (FWV) have the so called non-holonomic kinematics that results in a non-linear mapping between the control input and the states. Trajectory optimization with such motion models take the form of a challenging non-linear programming problem (NLP) [4]. Nevertheless, techniques like gradient descent [5], [6], sequential quadratic programming (SQP) [7] and primal-dual interior point implemented in open-source packages like SciPy [8], IPOPT [9] etc., can be employed to obtain a locally optimal solution.

The intractability of an NLP can be countered to a large extent by exploiting the niche mathematical structures of the problem. For example, consider the problem of computing a rank-1 approximation of a given matrix. Instead of treating it as an NLP and applying gradient descent on it, we can exploit the bi-convex structure of the problem and employ algorithms like alternating minimization which leads to closed form solution [10]. The proposed work brings similar perspective to non-holonomic trajectory optimization.

1. Institute of Technology, University of Tartu. 2. Robotics Research Center, IIIT-Hyderabad, India. This work was in part funded by the European Social Fund through IT Academy of Estonia Grant (SLTTI19605T) to the first author, Estonian Centre of Excellence in IT (EXCITE) funded by the European Regional Development Fund, base funding of University of Tartu Grant 20907, and UT-Bolt Consortium on Autonomous Driving (LLTAT19380).

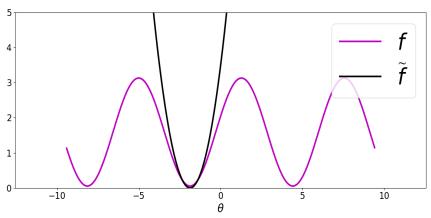

Fig. 1. One of the building blocks of the proposed trajectory optimizer. The magenta surface represents a non-convex function $f(\theta)$ which has multiple local minima. However, it is possible to derive a convex surrogate $\tilde{f}(\theta)$ (shown in black) that extracts one of the local minima of f.

Contributions and Overview of the Proposed Optimizer: In this paper, we reduce non-holonomic trajectory optimization to a problem of solving a sequence of globally valid convex QPs. The term "globally valid" is the key here as it differentiates the proposed optimizer with existing SQP based approaches like [11] (see Section III for details). At the core of our formulation is a simple insight that although the non-holonomic motion model is highly non-linear in the space of velocity v_t and heading angle ϕ_t, it is bi-affine in the space of v_t and $(\cos\phi_t, \sin\phi_t)$. Furthermore, we show that it is possible to induce similar structure in the collision avoidance constraints as well. Using these insights, we reformulate non-holonomic trajectory optimization in terms of a set of bi-convex cost and constraint functions along with a non-linear penalty (see 10a). The bi-convex part can be interpreted as a computationally simpler relaxation for the non-holonomic trajectory optimization while the residual of the penalty quantifies how well its output obeys the non-holonomic behavior. The non-linear penalty can also be viewed as a projection operator which maps the output of the bi-convex part back to the space of feasible non-holonomic motions.

The primary complexity of the reformulated problem stems from minimizing the non-linear penalty. However, we show that by adopting an alternating (Gauss-Seidal) minimization approach [12], we naturally arrive at a convex surrogate for replacing the non-linear penalty (see Fig.1). We validate the proposed trajectory optimizer on common benchmarks encountered in autonomous driving. For FWV, we consider the problem of guiding it through obstacles to

a periodic path. In this context, we consider a class of non-linear cost and show how it can be accommodated within our trajectory optimizer while retaining the bi-convex structure. Furthermore, we show that on all the considered examples, our trajectory optimizer produces solution with similar cost as off the shelf solvers based on SQP but significantly outperforms those based on interior point methods. Furthermore, with respect to both these techniques, the proposed optimizer achieves more than 20x reduction in computation time (see Section IV).

II. Problem Formulation

A. Preliminaries

Symbols and Notations: We will use lower normal faced letters to represent scalars while bold-faced variants would represent vectors. Matrices are represented by bold upper case letters. We represent the time dependency of the variables through subscript t. The superscript T would represent transpose of a row/column vector or a matrix. With a slight abuse of notation, we use the same symbol f to represent a generic function at different places.

Bi-Convexity and Alternating Minimization:

Definition 1. *A function $f(\mathbf{z}_1, \mathbf{z}_2)$ is bi-convex (bi-affine) if for a given $\mathbf{z}_1$, f is convex (affine) in $\mathbf{z}_2$ and vice-versa.*

Alternating Minimization: Given a function $f(\mathbf{z}_1, \mathbf{z}_2)$, the alternating (or Gauss Seidel) minimization proceeds through the following iterates [12].

$$\mathbf{z}_1^{k+1} = \arg\min_{\mathbf{z}_1} f(\mathbf{z}_1, \mathbf{z}_2^k), \mathbf{z}_2^{k+1} = \arg\min_{\mathbf{z}_2} f(\mathbf{z}_1^{k+1}, \mathbf{z}_2), \quad (1)$$

where, the superscript k refers to the value of the respective variable at iteration k. If function f is bi-convex in $\mathbf{z}_1, \mathbf{z}_2$, then each of the minimization in (1) is a convex optimization problem.

Motion Model: We consider the following motion model for non-holonomic car-like vehicles.

$$\dot{x}_t = v_t \cos\phi_t,\ \dot{y}_t = v_t \sin\phi_t,\ \ddot{v}_t = u_t^1,\ \dddot{\phi}_t = u_t^2. \quad (2a)$$

$$-v_t\kappa_{max} \leq \dot{\phi}_t \leq v_t\kappa_{max}, \forall t, \quad (2b)$$

where, x_t, y_t, v_t, ϕ_t represent the time-dependent Cartesian position, forward velocity and heading angle of the vehicle respectively. The control inputs (u_t^1, u_t^2) are taken as the linear and angular jerk. Inequality (2b) represents the curvature bound. The parameter κ_{max} is related to max steering angle δ_{max} and inter-axle distance l as $\kappa_{max} = \frac{\tan(\delta_{max})}{l}$.

B. Trajectory Optimization

To ensure smooth trajectories, we represent the position, heading and velocity variables in the following functional form.

$$x_t = \mathbf{p}_t\mathbf{c}_x, y_t = \mathbf{p}_t\mathbf{c}_y, \phi_t = \mathbf{p}_t\mathbf{c}_\phi, v = \mathbf{p}_{vt}\mathbf{c}_v, \quad (3)$$

where, $\mathbf{p}_t$ is a row vector formed with time-dependent cubic spline basis functions. Similarly, $\mathbf{p}_{v_t}$ is formed with quadratic polynomial basis functions. The variables $\mathbf{c}_x, \mathbf{c}_y, \mathbf{c}_v, \mathbf{c}_\phi$ are the coefficients associated with the basis functions. The higher-order derivatives can be computed through $\dot{\mathbf{p}}_t, \ddot{\mathbf{p}}_t$, etc. The chosen form ensures $\mathcal{C}^2$ continuity in all the trajectory variables with piece-wise constant linear and angular jerk. Using (3), the non-holonomic trajectory optimization can be framed as the following non-linear programming problem.

$$\arg\min_{\mathbf{c}_x, \mathbf{c}_y, \mathbf{c}_v, \mathbf{c}_\phi} f_{xy}(\mathbf{c}_x, \mathbf{c}_y) + f_v(\mathbf{c}_v) + f_\phi(\mathbf{c}_\phi). \quad (4a)$$

$$\mathbf{g}_t(\mathbf{c}_x, \mathbf{c}_y, \mathbf{c}_v, \mathbf{c}_\phi) = 0, \forall t. \quad (4b)$$

$$\mathbf{h}_v(\mathbf{c}_v) \leq 0. \quad (4c)$$

$$\mathbf{h}_\phi(\mathbf{c}_\phi) \leq 0. \quad (4d)$$

$$\mathbf{h}_{curv}(\mathbf{c}_\phi, \mathbf{c}_{vel}) \leq 0. \quad (4e)$$

$$\mathbf{h}_{o_t}^j(\mathbf{c}_x, \mathbf{c}_y, \mathbf{c}_\phi) \leq 0, \forall t, \forall j. \quad (4f)$$

The function $\mathbf{g}_t$ is obtained by substituting (3) into the first two equations of (2a). The cost function f_{xy} model costs which explicitly depend on x_t, y_t and their derivatives such as goal-reaching, lateral acceleration, etc. and has a convex quadratic form. The cost terms f_v, f_ϕ model regularization on velocity, acceleration and jerk and have the same form as f_{xy}. The functions $\mathbf{h}_v, \mathbf{h}_\phi, \mathbf{h}_{curv}$ in (4c)-(4e) are affine and model the bounds on linear and angular velocities, accelerations, jerk and curvature. Inequalities (4f) represent the collision avoidance constraints with respect to the j^{th} obstacle.

The computational bottleneck of (4a)-(4f) stems from the fact that $\mathbf{g}_t$ is highly non-linear comprising of trigonometric sine and cosine function of $\mathbf{c}_\phi$. Depending on the obstacle geometry, the same non-linearity may also manifest in the function $\mathbf{h}_{o_t}^j$. Existing works solve (4a)-(4f) as a standard NLP using algorithms like SQP. In the next section, we present a fundamentally different approach where we exploit the mathematical structures in $\mathbf{g}_t$ and $\mathbf{h}_{o_t}^j$ to reduce (4a)-(4f) to a set of globally valid convex QPs.

III. Main Results

In this section, we present a bi-convex approximation of (4a)-(4f). We begin by describing the various building blocks.

A. Convex Surrogate

The discussions presented here builds upon our prior work [13], [14]. Consider the following two optimization problems in term of variable θ for the given constants w_s and w_c.

$$\min_\theta f,\ f = (\cos(\theta) - w_c)^2 + (\sin(\theta) - w_s)^2 \quad (5a)$$

$$\min_\theta \widetilde{f},\ \widetilde{f} = \left(\theta - \arctan 2\frac{w_s}{w_c}\right)^2 \quad (5b)$$

The loss surface of f is shown in Fig.1 (in magenta) and consists of multiple minima, although all of them are of equal value. On the other hand, $\widetilde{f}$ is a linear least-squares function with a single global minima. Interestingly, $\bigtriangledown f = 0$ and $\bigtriangledown \widetilde{f} = 0$ have a common solution given by $\theta = \arctan 2(\frac{w_s}{w_c})$. Consequently, optimization (5a) and (5b) share a common minima. An alternative interpretation is that $\widetilde{f}$ acts as a convex surrogate for f. From anywhere in the θ space, one can follow the loss surface of $\widetilde{f}$ to extract one of the minima of f. Now, the very nature of $\arctan 2$ function would ensure that $\widetilde{f}$ always extracts the minima from the region $|\theta| \leq \pi$. However, this is a minor limitation, if we assume that θ actually represents an angle, in which case $|\theta| \leq \pi$ is large enough to cover the whole rotation range.

B. Mathematical Structure of $\boldsymbol{h}^j_{o_t}$

We follow [4] and approximate the geometry of the ego-vehicle through multiple circles and obstacles as axis rotated ellipses. Let the i^{th} circle center be (x^i_t, y^i_t) with $x^i_t = x_t + r_i\cos\phi_t, y^i_t = y_t + r_i\sin\phi_t$ for some constant r_i. Consequently, the obstacle avoidance constraint function between the i^{th} circle of the ego-vehicle and the j^{th} obstacle ellipse can be written as

$$h^{ij}_{o_t} = -\frac{1}{d_x^2}((x^i_t - x_t^{o_j})\cos(\alpha_t^{o_j}) + (y^i_t - y_t^{o_j})\sin(\alpha_t^{o_j}))^2 - \frac{1}{d_y^2}(-(x^i_t - x_t^{o_j})\sin(\alpha_t^{o_j}) + (y^i_t - y_t^{o_j})\cos(\alpha_t^{o_j}))^2 + 1, \quad (6)$$

where, $h^{ij}_{o_t}$ represents the i^{th} element of $\mathbf{h}^j_{o_t}$. The terms $x_t^{o_j}, y_t^{o_j}, \alpha_t^{o_j}$ respectively represent the center and axis angle of the j^{th} obstacle ellipse at time t . The constants d_x, d_y are the length of its major and minor axis respectively inflated by the radius of the i^{th} circle of the ego-vehicle. Function $h^{ij}_{o_t}$ is highly non-linear with respect to ϕ_t but is concave (reverse convex) in terms of variable x_t, y_t and $(\cos\phi_t, \sin\phi_t)$. Thus, it can be replaced with its affine approximation (7) obtained by Taylor series expansion around some guess $(\hat{x}_t, \hat{y}_t, \cos\hat{\phi}_t, \sin\hat{\phi}_t)$. Irrespective of the expansion point, the satisfaction of (7) ensures the satisfaction of original quadratic inequality (6) and consequently, the former acts as a convex (over) approximation of the latter [2].

$$h^{ij}_{o_t} = \frac{\partial h^{ij}_{o_t}}{\partial x_t}x_t + \frac{\partial h^{ij}_{o_t}}{\partial y_t}y_t + \frac{\partial h^{ij}_{o_t}}{\partial \cos(\phi_t)}\cos(\phi_t) + \frac{\partial h^{ij}_{o_t}}{\partial \sin(\phi_t)}\sin(\phi_t) + f^{ij}_{o_t}(\hat{x}_t, \hat{y}_t, \cos(\hat{\phi}_t), \sin(\hat{\phi}_t)). \quad (7)$$

Note that we take the partial derivative with respect to $(\cos\phi_t, \sin\phi_t)$ and not ϕ_t. The partial derivatives and $f^{ij}_{o_t}$ are scalar constants whose numerical values are determined by the expansion point $(\hat{x}_t, \hat{y}_t, \cos\hat{\phi}_t, \sin\hat{\phi}_t)$.

It is worth pointing out that the substitution of concave collision avoidance constraints with more conservative affine approximation has been exploited in many works but for robots with affine motion model (e.g. quadrotors [3]). However, this result until now had limited utility in the context of non-holonomic trajectory optimization due to the presence of trigonometric variables in (6). As we have shown, the key idea is to compute the affine approximation, not in the space of ϕ_t but rather $(\cos\phi_t, \sin\phi_t)$ which we next show fits perfectly into the proposed trajectory optimizer.

C. Reformulating and solving (4a)-(4f)

The reformulations presented here consists of two key steps. First, we introduce the following change of variables

$$\theta_t = \mathbf{p}_t\mathbf{c}_\phi, w_{c_t} = \cos\theta_t, w_{s_t} = \sin\theta_t \quad (8)$$

Second, we note that the functions $\mathbf{g}_t$ and $\mathbf{h}^j_{o_t}$ have the following general structure.

$$\mathbf{g}_t = \mathbf{A}_g(\mathbf{c}_v)\begin{bmatrix} w_{c_t} \\ w_{s_t}\end{bmatrix} + \mathbf{b}_g(\mathbf{c}_x, \mathbf{c}_y) \quad (9a)$$

$$\mathbf{h}^j_{o_t}(\mathbf{c}_x, \mathbf{c}_y, w_{c_t}, w_{s_t}) = \mathbf{A}^j_{xy}\begin{bmatrix}\mathbf{c}_x \\ \mathbf{c}_y\end{bmatrix} + \mathbf{A}^j_\phi\begin{bmatrix} w_{c_t} \\ w_{s_t}\end{bmatrix} + \mathbf{f}^j_{o_t} \quad (9b)$$

Representation of $\mathbf{g}_t$ in the form given by (9a) is straightforward and follows directly from substituting (3) in (2a) and subsequently performing the change of variables presented in (8). Similarly, $\mathbf{h}^j_{o_t}$ as (9b) directly follows from (3), (7) and (8). The matrices $\mathbf{A}^j_{xy}, \mathbf{A}^j_\phi$ and vector $\mathbf{f}^j_{o_t}$ are constants determined by the partial derivatives in (7) and $\mathbf{p}_t$.

In (9a), $\mathbf{A}_g$ is a matrix whose elements are affine functions of $\mathbf{c}_v$. Similarly $\mathbf{b}_g$ is a vector whose elements are affine functions of $\mathbf{c}_x, \mathbf{c}_y$. Clearly, (9a) represents a bi-affine function in terms of two block of variables $(\mathbf{c}_x, \mathbf{c}_y, \mathbf{c}_v)$ and (w_{c_t}, w_{s_t}). Similarly $\mathbf{h}^j_{o_t}$ is convex in the space of $(\mathbf{c}_x, \mathbf{c}_y)$ and w_{c_t}, w_{s_t}. Using (8) and (9a)-(9b), we reformulate (4a)-(4f) in the following form.

$$\min_{\mathbf{c}_v, \mathbf{c}_x, \mathbf{c}_y, \mathbf{c}_\phi, w_{c_t}, w_{s_t}, \theta_t} \mathcal{L}(\mathbf{c}_v, \mathbf{c}_x, \mathbf{c}_y, \mathbf{c}_\phi, w_{c_t}, w_{s_t}) + \sum_t \overbrace{\rho_c \left\| \begin{bmatrix} w_{c_t} - \cos\theta_t \\ w_{s_t} - \sin\theta_t \end{bmatrix}\right\|_2^2}^{consensus} + \boldsymbol{\lambda_{w_t}}^T \begin{bmatrix} w_{c_t} \\ w_{s_t}\end{bmatrix} + \lambda_{\theta_t}\theta_t \quad (10a)$$

$$\theta_t = \mathbf{p}_t\mathbf{c}_\phi \quad (10b)$$

$$\mathbf{h}_v, \mathbf{h}_\phi, \mathbf{h}_{curv} \leq 0 \quad (10c)$$

$$\mathbf{h}^j_{o_t}(\mathbf{c}_x, \mathbf{c}_y, w_{c_t}, w_{s_t}) \leq 0 \quad (10d)$$

$$\mathcal{L}(\mathbf{c}_v, \mathbf{c}_x, \mathbf{c}_y, \mathbf{c}_\phi, w_{c_t}, w_{s_t}) = f_{xy}(\mathbf{c}_x, \mathbf{c}_y) + f_v(\mathbf{c}_v) + f_\phi(\mathbf{c}_\phi) + \sum_t \boldsymbol{\lambda}^T_{g_t}(\mathbf{g}_t(\mathbf{c}_x, \mathbf{c}_y, \mathbf{c}_v, w_{c_t}, w_{s_t})) + \rho_g\|\mathbf{g}_t(\mathbf{c}_x, \mathbf{c}_y, \mathbf{c}_v, w_{c_t}, w_{s_t})\|_2^2$$

As can be seen, we have reformulated the equality constraints (4b) as a quadratic penalty along with a Lagrangian term with multiplier $\boldsymbol{\lambda}_{g_t}$. The cost function also has an additional non-linear penalty which ensures the consensus between w_{c_t} and $\cos\theta_t$ and between w_{s_t} and $\sin\theta_t$. The Lagrangian terms with $\boldsymbol{\lambda}_{w_t}$ and λ_{θ_t} are added to ensure that the consensus residual is driven very close to zero [13]. There is also an additional equality constraint which forces agreement between θ_t and $\mathbf{p}_t\mathbf{c}_\phi$. Note that $\mathcal{L}$ is bi-convex in two blocks of variable $(\mathbf{c}_v, \mathbf{c}_x, \mathbf{c}_y, \mathbf{c}_\phi)$ and (w_{c_t}, w_{s_t}).

The form presented in (10a)-(10c) has a simple intuitive interpretation. If we disregard, the difficult consensus terms in the cost function, we get a bi-convex optimization problem which can be efficiently solved through alternating minimization algorithm. However, the resulting trajectory will not obey the non-holonomic kinematics. But as we reduce the consensus residuals, the resulting trajectory would start approaching the non-holonomic behavior.

On the first look, (10a)-(10d) resemble a trivial re-phrasing of (4a)-(4f) and with no clear computational advantage. In particular, the non-linear consensus penalty makes (10a)-(10d) potentially as difficult as (4a)-(4e). Thus, to explicitly highlight the utility of the former, we next apply alternating minimization algorithm on it.

Given an initial guess for $\mathbf{c}^k_\phi, \mathbf{c}^k_v, \mathbf{c}^k_x, \mathbf{c}^k_y, \theta^k_t$ at $k = 0$, the solution of (10a)-(10d) proceeds through the iterates presented in Algorithm 1, where the superscript k represents the value of the respective variable at iteration k. Steps (13a)-(13c) are similar to (1) and involves solving convex QPs. The first line of minimization (13d) represents a difficult non-convex optimization. But interestingly, for the given $w^{k+1}_{c_t}, w^{k+1}_{s_t}$, the non-linear penalty has the same form as (5a)

and thus we can replace it with a convex surrogate similar to (5b). We follow [15] for updating the Lagrange multipliers λ_{g_t} based on residual of $\mathbf{g}_t$. The update of $\boldsymbol{\lambda}_{w_t}, \lambda_{\theta_t}$ are derived from our prior work [13] in the following form.

$$\boldsymbol{\lambda}_{w_t} = \boldsymbol{\lambda}_{w_t} + \begin{bmatrix} w_{c_t}^{k+1} - \cos(\theta_t^{k+1}) \\ w_{s_t}^{k+1} - \sin(\theta_t^{k+1}) \end{bmatrix} \tag{11}$$

$$\lambda_{\theta_t} = \lambda_{\theta_t} + (\theta_t^{k+1} - \arctan 2 \frac{w_{st}^{k+1}}{w_{c_t}^{k+1}}) \tag{12}$$

The optimizations (13c) and (13d) can be understood as back and forth projection between the set containing w_{c_t}, w_{s_t} and θ_t. Such alternating projections are found to converge even in the non-convex case [16]. Refer to our prior work [13] (Fig. 1) for visualization of the projection process. We also empirically validate the convergence in Section IV by showing a decrease in consensus residual with iteration.

Comparisons with SQP: The Algorithm 1 has a fundamental difference with algorithms like SQP which at the most basic level also involve solving a sequence of convex QPs. In the context of optimization (4a)-(4f), SQP would proceed by Taylor series expansion of $\mathbf{g}_t, \mathbf{h}_{o_t}^j$ with respect to ϕ_t to obtain a convex QP. Since, both these functions are highly nonlinear in terms of ϕ_t, the obtained QP will hold only in the vicinity of the expansion point. This in turn, leads to a requirement of incorporating line-search or so-called trust-region constraints (see [11]) which forces the solution at each iteration to be in the vicinity of the current expansion point. In contrast, we linearized $\mathbf{h}_{o_t}^j$ around $(\cos\phi_t, \sin\phi_t)$ and thus obtained a globally valid convex (over) approximation. For $\mathbf{g}_t$, we did not perform any linearization. Finally, we by-pass the non-convexity of $(\cos\phi_t, \sin\phi_t)$ through smart reformulations and globally valid convex surrogate (5b). As a result, Algorithm 1 requires no line search or trust region constraints and thus there is no restriction on how big a step, the QPs can take towards the optimal solution.

D. Extension to FWV

The planar motion of a FWV is governed by the non-holonomic motion model presented in (2a). However, the curvature bounds are bi-affine, in contrast to the affine form associated with non-holonomic car-like vehicles [17]. Existing works like [17] typically fix forward velocity v_t of FWV to a constant value to reduce bi-affine curvature bounds to an affine form. However, such an approach reduces the space of feasible motions. In contrast, the proposed optimizer does not require such simplifying assumptions as it can directly handle the bi-convex form by alternately optimizing over the linear and angular velocities (see (13a) -(13d)).

Goal Set: Consider the following cost function where x_c, y_c correspond to the center of a circular or a elliptical path parametrized in terms of arc angle γ as $x_{path} = x_c + a\cos(\gamma), y_{path} = y_c + b\sin(\gamma)$

$$\begin{aligned} &f_{goalset}(x_t, y_t, \gamma) = (x_t - x_{path})^2 + (y_t - y_{path})^2, t = t_f \\ &\Rightarrow f_{goalset}(\mathbf{c}_x, \mathbf{c}_y, \gamma) = (\mathbf{p}_t\mathbf{c}_x - x_{path})^2 + (\mathbf{p}_t\mathbf{c}_y - y_{path})^2 \end{aligned} \tag{14}$$

The cost function defines a set of goal positions that a FWV can take at final time t_f, thus allowing the FWV to choose a point of convergence on the given path based on trajectory smoothness, obstacle avoidance etc. Note that the cost function is non-linear with respect to variable γ but convex with respect to $(\cos\gamma, \sin\gamma)$. Thus, we follow the process established in the previous section and perform the change of variables $w_c^\gamma = \cos(\gamma)$ and $w_s^\gamma = \sin(\gamma)$. The cost function is subsequently reformulated in the following manner.

$$\begin{aligned} f_{goalset}(\mathbf{c}_x, \mathbf{c}_y, w_c^\gamma, w_s^\gamma) + \sum_t \overbrace{\rho_\gamma \| \begin{bmatrix} w_c^\gamma - \cos(\gamma) \\ w_s^\gamma - \sin(\gamma) \end{bmatrix} \|_2^2}^{consensus} \\ + \boldsymbol{\lambda}_{w^\gamma}^T \begin{bmatrix} w_c^\gamma \\ w_s^\gamma \end{bmatrix} + \lambda_\gamma \gamma \end{aligned} \tag{15}$$

The cost (15) can now be integrated in Algorithm 1. Two new minimization steps similar to (13c) and (13d) would result for computing w_c^γ, w_s^γ and γ respectively.

Tangent Cost: We can also construct a cost to ensure that at the point of convergence, the heading angle of the FWV is tangential to the given path. That is,

$$f_{tangent} = (\dot{x}_t\cos\gamma + \dot{y}_t\sin\gamma)^2 = (\dot{x}_t w_c^\gamma + \dot{y}_t w_s^\gamma)^2, t = t_f \tag{16}$$

It can be seen that the cost (16) has the same form as $f_{goalset}$.

IV. Simulation and Benchmarking

Setup: We prototyped the proposed optimizer in Python using either QuadProg [18] or CVXOPT [19] as our QP solver. We also implemented optimization (4a)-(4f) using SQP method implemented in SciPy-SLSQP [8] and interior-point method implemented in IPOPT [9] using Autograd [20] to compute the gradients of the cost and constraint functions. All optimizers were run on a 2.60 GHz laptop with 32 GB RAM, i7-9750H CPU on a single core set up. For each benchmark considered, we generated 11 different variants by perturbing the initial positions, heading and forward velocity. The planning horizon was $15s$ for the autonomous driving benchmarks and $10s$ for the example involving FWV and was further discretized into 50 steps.

Benchmark 1: In this benchmark shown in Fig.2(a), the ego-vehicle (shown in red) is required to converge behind the leader vehicle (shown in blue) with zero relative velocity while avoiding the dynamic obstacles (shown in black). This benchmark was chosen because it requires the ego-vehicle to perform overtaking maneuvers and for some initial y position of the ego-vehicle also entails a lane change action.

The cost function employed consisted of a cost on the final position (as determined by the leader vehicle's trajectory in the planning horizon) and smoothness cost consisting of penalties on lateral acceleration $(\ddot{y})$, linear $(\ddot{v}_t)$ and angular $(\dddot{\phi}_t)$ jerk. The relative velocity requirement was modeled as affine equalities on variables $\mathbf{c}_v, \mathbf{c}_\phi$. We also considered additional affine inequalities on variables $\mathbf{c}_x, \mathbf{c}_y$ to model the lane boundaries.

The output trajectories from two different initial positions are shown in Fig.2(a). Fig2(b) provides an empirical validation for the convergence of Algorithm 1 by showing decrease

Algorithm 1 Alternating Minimization for solving (10a)-(10d)

1: Initialize $\mathbf{c}_\phi^k, \mathbf{c}_v^k, \mathbf{c}_x^k, \mathbf{c}_y^k, \theta_t^k$.
2: **while** $k \leq maxiter$ **do**

Use $\mathbf{c}_x^k, \mathbf{c}_y^k, \mathbf{c}_\phi^k$ to compute $(\hat{x}_t, \hat{y}_t, \cos\hat{\phi}_t, \sin\hat{\phi}_t)$ and consequently $\mathbf{A}_{xy}^j, \mathbf{A}_\phi^j, \mathbf{f}_{o_t}^j$, Update $w_{c_t}^k = \cos\theta_t^k, w_{s_t}^k = \sin\theta_t^k$.

$$\mathbf{c}_v^{k+1} = \arg\min_{\mathbf{c}_v} \mathcal{L}(\mathbf{c}_v, \mathbf{c}_x^k, \mathbf{c}_y^k, \mathbf{c}_\phi^k, w_{c_t}^k, w_{s_t}^k), \mathbf{h}_v(\mathbf{c}_v) \leq 0, \mathbf{h}_{curv}(\mathbf{c}_v, \mathbf{c}_\phi^k) \leq 0. \quad (13a)$$

$$\mathbf{c}_x^{k+1}, \mathbf{c}_y^{k+1} = \arg\min_{\mathbf{c}_x, \mathbf{c}_y} \mathcal{L}(\mathbf{c}_v^{k+1}, \mathbf{c}_x, \mathbf{c}_y, \mathbf{c}_\phi^k, w_{c_t}^k, w_{s_t}^k), \mathbf{h}_{o_t}^j(\mathbf{c}_x, \mathbf{c}_y, w_{c_t}^k, w_{s_t}^k) \leq 0. \quad (13b)$$

Update $\mathbf{A}_{xy}^j, \mathbf{A}_\phi^j, \mathbf{f}_{o_t}^j$ using $\mathbf{c}_x^{k+1}, \mathbf{c}_y^{k+1}$.

$$w_{c_t}^{k+1}, w_{s_t}^{k+1} = \arg\min_{w_{c_t}, w_{s_t}} \mathcal{L}(\mathbf{c}_v^{k+1}, \mathbf{c}_x^{k+1}, \mathbf{c}_y^{k+1}, \mathbf{c}_\phi^k, w_{c_t}, w_{s_t}) + \rho_c \| \begin{bmatrix} w_{c_t} - \cos\theta_t^k \\ w_{s_t} - \sin\theta_t^k \end{bmatrix} \|_2^2 + \boldsymbol{\lambda_{w_t}}^T \begin{bmatrix} w_{c_t} \\ w_{s_t} \end{bmatrix}, \mathbf{h}_{o_t}^j(\mathbf{c}_x^{k+1}, \mathbf{c}_y^{k+1}, w_{c_t}, w_{s_t}) \leq 0. \quad (13c)$$

$$\theta_t^{k+1}, \mathbf{c}_\phi^{k+1} = \arg\min_{\theta_t, \mathbf{c}_\phi} \mathcal{L}(\mathbf{c}_v^{k+1}, \mathbf{c}_x^{k+1}, \mathbf{c}_y^{k+1}, \mathbf{c}_\phi, w_{c_t}^{k+1}, w_{s_t}^{k+1}) + \sum_t \rho_c \| \begin{bmatrix} w_{c_t}^{k+1} - \cos\theta_t \\ w_{s_t}^{k+1} - \sin\theta_t \end{bmatrix} \|_2^2 + \lambda_{\theta_t}(\theta_t)$$

$$\theta_t = \mathbf{p}_t \mathbf{c}_\phi, \mathbf{h}_\phi(\mathbf{c}_\phi) \leq 0, \mathbf{h}_{curv}(\mathbf{c}_\phi, \mathbf{c}_v^{k+1}) \leq 0.$$

$$\Rightarrow \theta_t^{k+1}, \mathbf{c}_\phi^{k+1} = \arg\min_{\theta_t, \mathbf{c}_\phi} \mathcal{L}(\mathbf{c}_v^{k+1}, \mathbf{c}_x^{k+1}, \mathbf{c}_y^{k+1}, \mathbf{c}_\phi, w_{c_t}^{k+1}, w_{s_t}^{k+1}) + \sum_t \rho_c (\theta_t - \arctan 2 \frac{w_{c_t}^{k+1}}{w_{s_t}^{k+1}})^2 + \lambda_{\theta_t}(\theta_t)$$

$$\theta_t = \mathbf{p}_t \mathbf{c}_\phi, \mathbf{h}_\phi(\mathbf{c}_\phi) \leq 0, \mathbf{h}_{curv}(\mathbf{c}_\phi, \mathbf{c}_v^{k+1}) \leq 0. \quad (13d)$$

3: **end while**

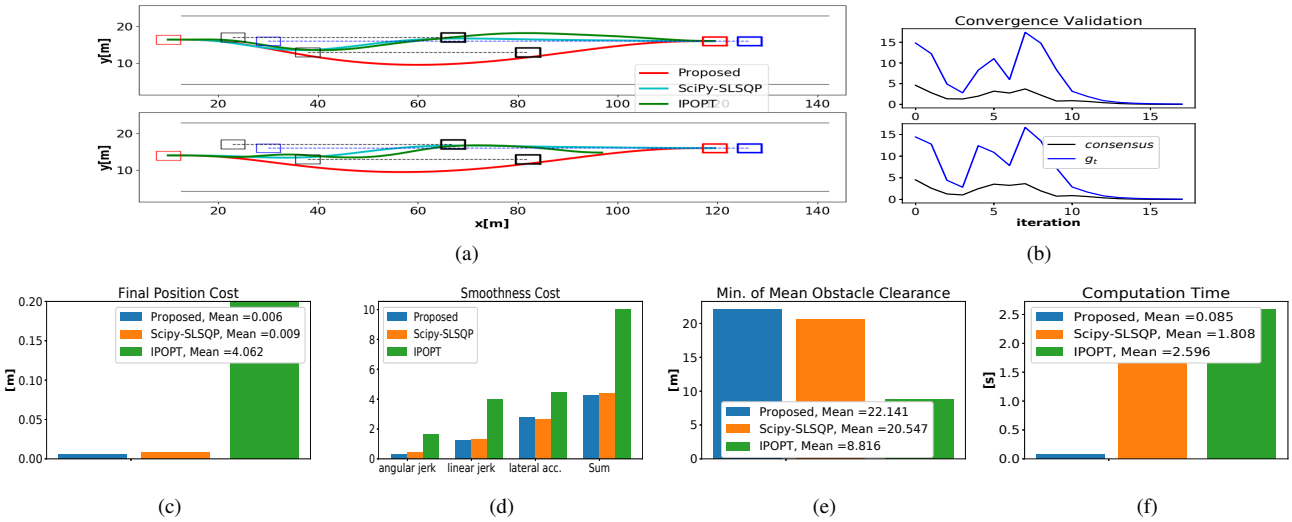

Fig. 2. (a) The ego-vehicle (red rectangle) needs to converge behind the leader vehicle (blue rectangle) with zero relative velocity while avoiding the dynamic obstacles (black rectangles). The initial positions of the vehicles are shown in the lighter shade while the bold variants represent the final position. The final position of the ego-vehicle is only shown with respect to the trajectory obtained with the proposed optimizer. The top and bottom plots show the trajectories for two initial positions of the ego-vehicle. (b) Residual of the consensus and $\|\mathbf{g}_t\|_2$ with iterations for the two trajectories in (a) obtained with the proposed optimizer. (c)-(f) Quantitative comparisons between the proposed optimizer, SciPy-SLSQP and IPOPT across different metrics. Note lateral acceleration is $\ddot{y}_t$, linear and angular jerk are $\dddot{v}_t$ and $\dddot{\phi}_t$ respectively.

in residual of consensus term and $\|\mathbf{g}_t\|_2^2$ with iterations. The quantitative results are summarized in Fig.2(c)-2(f). Both Scipy-SLSQP and the proposed optimizer produced similar final position and smoothness cost which were significantly better than those resulting from IPOPT. both SciPy-SLSQP and IPOPT produces trajectories in the same homoptopy which went between the obstacles. In contrast, the trajectories resulting from the proposed optimizer consistently chose a homotopy with higher clearance from dynamic obstacles. This is further validated in Fig.2(e) which shows the minimum of the mean obstacle clearance observed across different trajectories. Fig.2(f) shows our most important result: the computation time of the proposed trajectory optimizer is more than 20 and 30 times less than SciPy-SLSQP and IPOPT respectively. Note that the timing results for the latter two optimizer do not include the Autograd overhead.

Benchmark 2: This benchmark is shown in Fig.3(a) and is similar to one shown in Fig.2(a) with the same cost function. The difference here is that the ego-vehicle is required to converge to the front of the leader vehicle with zero relative velocity. The results are summarized in Fig.3(a)-(3(f)). The proposed optimizer again outperforms both Scipy-SLSQP and IPOPT computation time by a factor of almost 19 and 16 respectively. Note that the increased computation time in this case with respect to benchmark 1 is due to the fact that the ego-vehicle needs to consider collision avoidance with the leader vehicle while overtaking it. Thus, there are effectively three dynamic obstacles in this benchmark.

Benchmark 3: This benchmark is shown in Fig.4(a) and requires an FWV to converge to a circular path (shown in magenta) while avoiding circular obstacles. As common in existing works, we also model the footprint of the FWV

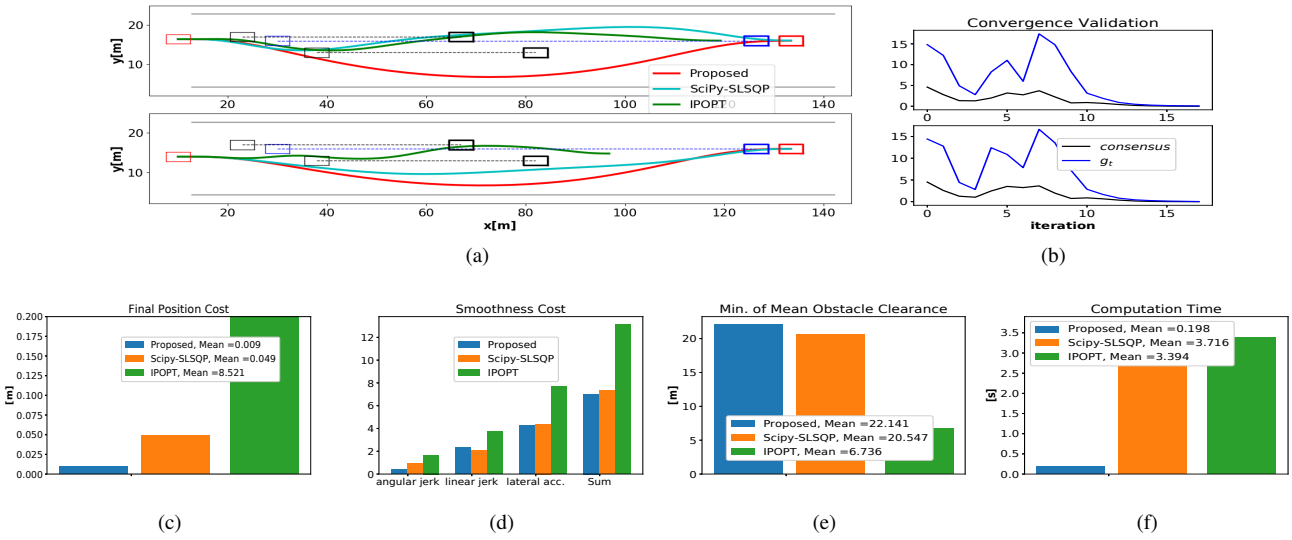

Fig. 3. (a) Similar benchmark as Fig.3 with the same plotting conventions. The difference here is that the ego-vehicle needs to converge in front of the leader vehicle with zero relative velocity. (b) Convergence validation for the two trajectories shown in (a) obtained with the proposed optimizer. (c)-(f) Quantitative comparisons between the proposed optimizer, SciPy-SLSQP and IPOPT across different metrics.

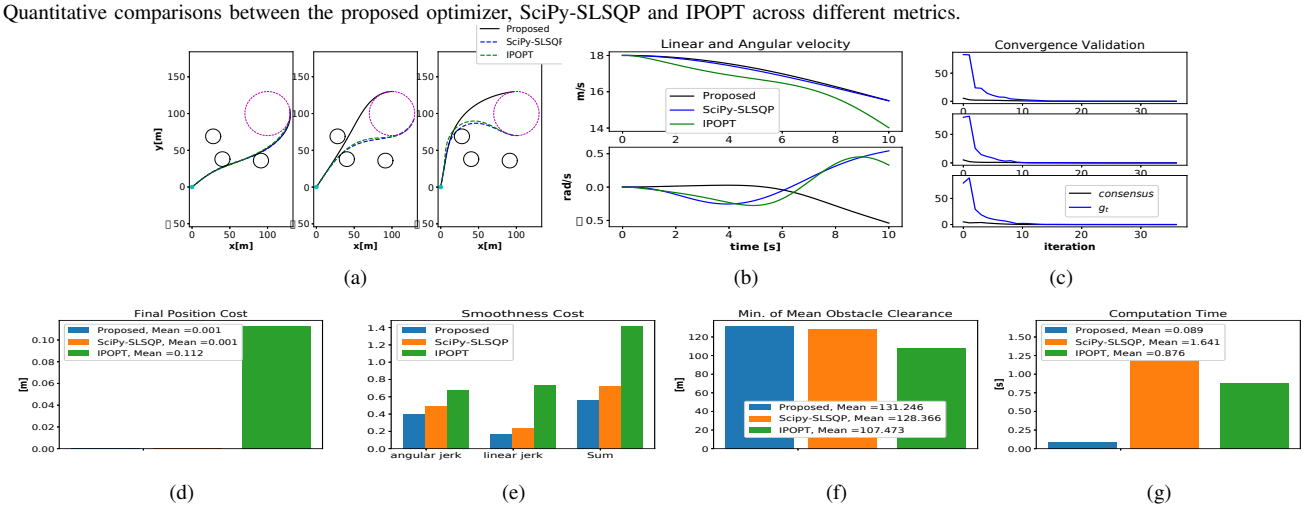

Fig. 4. (a) FWV starting from an initial point (shown in cyan) needs to converge to a circular path (shown in magenta) with a heading tangential to the path and while avoiding the static obstacles (shown in black). (b) Linear and Angular velocity profiles obtained for the third trajectory shown in (a) with different optimizers. (c) Convergence validation for the three trajectories shown in (a) obtained with the proposed optimizer. (d)-(g) Quantitative comparisons across different metrics.

as a circle. The cost function employed in this benchmark consisted of $f_{goalset}$ and $f_{tangent}$ defined in (14) and (16) respectively and smoothness cost in the form of l_2 norm penalties on linear $(\ddot{v}_t)$ and angular jerk $(\dddot{\phi}_t)$.

The output trajectories for three different initial heading profile are shown in Fig.4(a). Fig.4(b) verifies the smoothness in the linear and angular velocity profiles for the third trajectory shown in Fig.4(a). Fig.4(c) shows the progress of the consensus and non-holonomic constraint residuals with iteration for the three trajectories shown in Fig.4(a). The quantitative results are summarized in Fig.4(d)-4(g). IPOPT performed much better in this benchmark in terms of final position and smoothness cost although still worse than the proposed optimizer and the SciPy-SLSQP. Just like previous benchmarks, we observed slightly higher obstacle clearance for the trajectories obtained from the proposed optimizer (Fig.4(f)). The computation times (Fig.4(g)) again show the proposed optimizer outperforming SciPy-SLSQP and IPOPT, in this case, by a factor of 18 and 10 respectively.

V. Discussions and Future Work

In this paper, we showed that non-holonomic trajectory optimization has some niche mathematical structures which have been overlooked in the existing literature. We proposed some intelligent reformulations around these structures and reduced non-holonomic trajectory optimization to a problem of solving a sequence of globally valid convex QPs. We performed extensive comparisons with state of the art open-source optimizers and showed either similar or improved performance in terms of achieved optimal cost but with more than 20x reduction in computation time.

The proposed optimizer can be immediately extended to accommodate motion model of cars with side-slip or tyre force dynamics since they have the same structure of being bi-affine in the space of velocity and cosine and sine of heading, side slip and steering angles [21]. Similar structures can be also be found in 3D motion model for FWV [17].

REFERENCES

[1] D. Mellinger and V. Kumar, "Minimum snap trajectory generation and control for quadrotors," in *2011 IEEE International Conference on Robotics and Automation*. IEEE, 2011, pp. 2520–2525.

[2] T. Lipp and S. Boyd, "Variations and extension of the convex–concave procedure," *Optimization and Engineering*, vol. 17, no. 2, pp. 263–287, 2016.

[3] F. Gao and S. Shen, "Quadrotor trajectory generation in dynamic environments using semi-definite relaxation on nonconvex qcqp," in *2017 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2017, pp. 6354–6361.

[4] W. Schwarting, J. Alonso-Mora, L. Pauli, S. Karaman, and D. Rus, "Parallel autonomy in automated vehicles: Safe motion generation with minimal intervention," in *2017 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2017, pp. 1928–1935.

[5] J. David, R. Valencia, R. Philippsen, P. Bosshard, and K. Iagnemma, "Gradient based path optimization method for autonomous driving," in *2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2017, pp. 4501–4508.

[6] K. Yang, S. Sukkarieh, and Y. Kang, "Adaptive nonlinear model predictive path tracking control for a fixed-wing unmanned aerial vehicle," in *AIAA Guidance, Navigation, and Control Conference*, 2009, p. 5622.

[7] C. Chen, Y. He, C. Bu, J. Han, and X. Zhang, "Quartic bézier curve based trajectory generation for autonomous vehicles with curvature and velocity constraints," in *2014 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2014, pp. 6108–6113.

[8] E. Jones, T. Oliphant, P. Peterson *et al.*, "Scipy: Open source scientific tools for python," 2001.

[9] 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, no. 1, pp. 25–57, 2006.

[10] Q. Li, Z. Zhu, and G. Tang, "Alternating minimizations converge to second-order optimal solutions," in *International Conference on Machine Learning*, 2019, pp. 3935–3943.

[11] 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.

[12] P. Jain, P. Kar *et al.*, "Non-convex optimization for machine learning," *Foundations and Trends® in Machine Learning*, vol. 10, no. 3-4, pp. 142–336, 2017.

[13] A. K. Singh, R. Ghabcheloo, A. Muller, and H. Pandya, "Combining method of alternating projections and augmented lagrangian for task constrained trajectory optimization," in *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2018, pp. 7568–7575.

[14] A. K. Singh, A. Ahonen, R. Ghabcheloo, and A. Muller, "Introducing multi-convexity in path constrained trajectory optimization for mobile manipulators," in *To appear at 2020 IEEE European Control Conference (ECC), related version available at arXiv preprint arXiv:1904.09780*, 2020.

[15] S. Boyd, N. Parikh, E. Chu, B. Peleato, J. Eckstein *et al.*, "Distributed optimization and statistical learning via the alternating direction method of multipliers," *Foundations and Trends® in Machine learning*, vol. 3, no. 1, pp. 1–122, 2011.

[16] A. S. Lewis, D. R. Luke, and J. Malick, "Local linear convergence for alternating and averaged nonconvex projections," *Foundations of Computational Mathematics*, vol. 9, no. 4, pp. 485–513, 2009.

[17] M. Owen, R. W. Beard, and T. W. McLain, "Implementing dubins airplane paths on fixed-wing uavs," *Handbook of Unmanned Aerial Vehicles*, pp. 1677–1701, 2015.

[18] D. Goldfarb and A. Idnani, "A numerically stable dual method for solving strictly convex quadratic programs," *Mathematical programming*, vol. 27, no. 1, pp. 1–33, 1983. [Online]. Available: https://pypi.org/project/quadprog/

[19] L. Vandenberghe, "The cvxopt linear and quadratic cone program solvers," *Online: http://cvxopt. org/documentation/coneprog. pdf*, 2010.

[20] D. Maclaurin, D. Duvenaud, and R. P. Adams, "Autograd: Effortless gradients in numpy," in *ICML 2015 AutoML Workshop*, vol. 238, 2015.

[21] J. Kong, M. Pfeiffer, G. Schildbach, and F. Borrelli, "Kinematic and dynamic vehicle models for autonomous driving control design," in *2015 IEEE Intelligent Vehicles Symposium (IV)*. IEEE, 2015, pp. 1094–1099.