Closed-Form Minkowski Sum Approximations for Efficient Optimization-Based Collision Avoidance

James Guthrie, Marin Kobilarov, Enrique Mallada

***Abstract*— Motion planning methods for autonomous systems based on nonlinear programming offer great flexibility in incorporating various dynamics, objectives, and constraints. One limitation of such tools is the difficulty of efficiently representing obstacle avoidance conditions for non-trivial shapes. For example, it is possible to define collision avoidance constraints suitable for nonlinear programming solvers in the canonical setting of a circular robot navigating around M convex polytopes over N time steps. However, it requires introducing $(2+L)MN$ additional constraints and LMN additional variables, with L being the number of halfplanes per polytope, leading to larger nonlinear programs with slower and less reliable solving time. In this paper, we overcome this issue by building closed-form representations of the collision avoidance conditions by outer-approximating the Minkowski sum conditions for collision. Our solution requires only MN constraints (and no additional variables), leading to a smaller nonlinear program. On motion planning problems for an autonomous car and quadcopter in cluttered environments, we achieve speedups of 4.0x and 10x respectively with significantly less variance in solve times and negligible impact on performance arising from the use of outer approximations.**

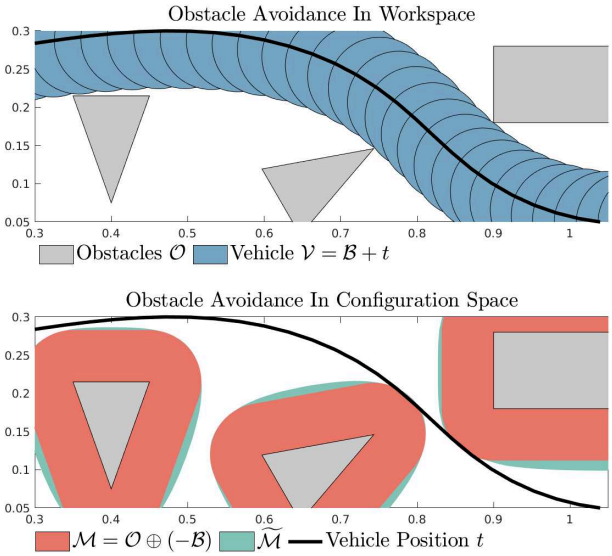

Fig. 1. Obstacle Avoidance in Workspace and Configuration Space

I. INTRODUCTION

Motion planning is a central task of most autonomous systems, including robots, drones, and autonomous vehicles. Of the many approaches to motion planning, techniques based on nonlinear programming (NLP) such as direct multiple shooting [1] and direct collocation [2] generally offer the most flexibility in regards to choice of objectives and constraints imposed. As high-quality NLP solvers and supporting automatic differentiation tools have become available, it has become feasible to utilize these optimization-based approaches for real-time motion planning or trajectory generation [3], [4].

Despite the flexibility that NLP solvers provide, it can be difficult to efficiently represent obstacle avoidance constraints. Due to their reliance on gradient and Hessian information, most NLP solvers require the objective and constraints to be twice continuously differentiable expressions. This presents a challenge for collision avoidance constraints which often cannot be represented in smooth closed-forms. We briefly review two viable approaches and discuss their advantages and limitations.

Distance Formulation: Collision avoidance can be viewed as ensuring the minimum distance between an obstacle $\mathcal{O}$ and a vehicle $\mathcal{V}$ is greater than zero. In robotics this would be classified as performing collision checking directly in the workspace [5] as shown in the upper plot of Figure 1. When the obstacle and the vehicle have convex shapes, the distance between these sets can be computed through convex optimization [6]. Using this formulation as a constraint leads to a bi-level nonlinear planning problem for which we lack reliable solvers. However, by leveraging strong duality, it is possible to reformulate the minimum-distance conditions into expressions amenable to a NLP solver [7], [8]. This method is advantageous in that it is simple to implement and provides an exact representation of the collision avoidance constraints. However, this is done at the expense of introducing new variables and constraints. In practice the method can be highly sensitive to the quality of the initial guess [9] and be unacceptably slow for real-time motion planning [4].

Minkowski Sum Formulation: Collision avoidance can alternatively be viewed through the lens of computational geometry as shown in the lower plot of Figure 1. Given the vehicle position $t \in \mathbb{R}^n$, and shapes $\mathcal{B}, \mathcal{O} \subset \mathbb{R}^n$ of vehicle and obstacle, respectively, collision avoidance can be posed as ensuring $t \notin \mathcal{M}$ where $\mathcal{M} = \mathcal{O} \oplus (-\mathcal{B})$, with $\oplus$ being the Minkowski sum operation [5]. In robotics this often referred to as the configuration-space (C-space) approach. Incorporating this as a constraint in an NLP solver would require a closed-form, smooth representation of the indicator function of this set. In general, this does

J. Guthrie and E. Mallada are with the Department of Electrical and Computer Engineering, Johns Hopkins University, Baltimore, MD 21218, USA. Email: {jguthri6, mallada}@jhu.edu.

M. Kobilarov is with the Department of Mechanical Engineering, Johns Hopkins University, Baltimore, MD 21218, USA. Email: marin@jhu.edu

not exist as the sets are semi-algebraic, involving multiple polynomial (in)equalities. A notable exception is the case of bodies whose boundary surface are smooth and admit both implicit and parametric representations [10], which includes spheres, ellipsoids [11], and more generally superquadrics [12]. However, many implicit surfaces do not admit a parametric representation and for others obtaining one is an open problem [13]. Additionally, this approach cannot address the practical case of non-smooth boundaries such as convex polytopes.

A. Contributions

In this work we propose efficient collision avoidance conditions based on closed-form, outer approximations $\widetilde{\mathcal{M}} \supseteq \mathcal{M}$ of the Minkowski sum. We focus on the important case in which the obstacle is a bounded, convex polytope and the vehicle is represented by Euclidean balls (possibly multiple). Building upon recent successes of sum-of-squares optimization for outer approximating semi-algebraic sets [14]–[17], we develop SOS programs for finding $\widetilde{\mathcal{M}}$. Figure 1 shows an example of the resulting outer approximations. We then use $\widetilde{\mathcal{M}}$ to perform optimization-based motion planning of an autonomous car and quadcopter navigating cluttered environments. Compared to the exact method [7], our approximate method solves 4x (car) and 10x (quadcopter) faster while introducing negligible conservatism arising from the use of outer approximations.

The rest of the paper is organized as follows. Section II reviews relevant aspects of convex sets, Minkowski sums, and SOS optimization. Section III defines the motion planning problem. Section IV poses the obstacle avoidance constraints using Minkowski sums and provides methods for outer approximating the set. Section V applies our approach to motion planning for an autonomous car and quadcopter. Section VI concludes the paper with a discussion of future directions.

II. PRELIMINARIES

We briefly review some basic properties of convex sets, Minkowski sums, and sum-of-squares polynomials. This is mostly done to setup our notation. The reader is referred to [6], [18], [19] for proofs and further details.

A. Set Definitions

Definition 1 (Convex Hull)**.** The convex hull of a set $\mathcal{B}$ is defined as: $\mathbf{conv}\ \mathcal{B} = \{\theta_1 x_1 + \ldots + \theta_k x_k \mid x_i \in \mathcal{B}, \theta_i \geq 0, i = 1, \ldots, k, \sum_{i=1}^{k} \theta_i = 1\}$ The convex hull has the property that it is the smallest convex set that contains $\mathcal{B}$. Let $\mathcal{C}$ be any convex set that contains $\mathcal{B}$. The following relation holds:

$$\mathcal{B} \subseteq \mathcal{C} \Leftrightarrow \mathbf{conv}\ \mathcal{B} \subseteq \mathcal{C} \tag{1}$$

Definition 2 (α-sublevel Set)**.** The α-sublevel set of a function $f : \mathbb{R}^n \to \mathbb{R}$ is: $B_\alpha = \{x | f(x) \leq \alpha\}$. We denote the boundary of the set as $\partial B_\alpha = \{x | f(x) = \alpha\}$.

Lemma 1. *Let $\mathcal{B}_\alpha$ be a convex set that is the α-sublevel set of a function $f : \mathbb{R}^n \to \mathbb{R}$. Then $\mathcal{B}_\alpha = \mathbf{conv}\ \partial\mathcal{B}_\alpha$.*

We use the notation $-\mathcal{B} = \{-b | b \in \mathcal{B}\}$ to represent the set $\mathcal{B}$ reflected about the origin. Note that $-\mathcal{B}$ is convex if and only $\mathcal{B}$ is convex.

Definition 3 (Polytope)**.** A polytope is defined as the solution set of j linear inequalities in $\mathbb{R}^n$. This set is convex by construction. We impose the additional requirement that the set is bounded. The linear inequalities give the halfspace representation

$$\mathcal{P} = \{x | Ax \leq b\} \tag{2}$$

where $A \in \mathbb{R}^{j \times n}, b \in \mathbb{R}^j$. Alternatively, the polytope can be represented by the convex hull of its k vertices

$$\mathcal{P} = \mathbf{conv}\ \{v_1, v_2, \ldots, v_k\} \tag{3}$$

where $v_i \in \mathbb{R}^n, i \in [k] := \{1, \ldots, k\}$.

B. Minkowski Sum Properties

Definition 4 (Minkowski Sum)**.** Given two sets $\mathcal{A}, \mathcal{B}$, their Minkowski sum is defined as follows:

$$\mathcal{A} \oplus \mathcal{B} = \{a + b | a \in \mathcal{A}, b \in \mathcal{B}\} \tag{4}$$

Lemma 2. *If A and B are convex sets then $A \oplus B$ is convex.*

Lemma 3. *For any sets $\mathcal{A}, \mathcal{B}$ the following equality holds:*

$$\mathbf{conv}\ (A \oplus B) = \mathbf{conv}\ (A) \oplus \mathbf{conv}\ (B) \tag{5}$$

C. Sum-of-Squares Optimization

For $x \in \mathbb{R}^n$, let $\mathbb{R}[x]$ denote the set of polynomials in x with real coefficients.

Definition 5 (Sum-of-Squares Polynomial)**.** A polynomial $p(x) \in \mathbb{R}[x]$ is a sum-of-squares (SOS) polynomial if there exists polynomials $q_i(x) \in \mathbb{R}[x], i = 1, \ldots, j$ such that $p(x) = q_1^2(x) + \ldots + q_j^2(x)$. We use $\sum[x]$ to denote the set of SOS polynomials in x. A polynomial of degree $2d$ is a SOS polynomial if and only if there exists a positive semi-definite matrix P (the Gram matrix) such that $p(x) = z(x)^T P z(x)$ where $z(x)$ is the vector of all monomials of x up to degree d [19].

Note that a polynomial being SOS is a sufficient condition for the polynomial to be non-negative (i.e. $p(x) \geq 0 \, \forall \, x$).

Definition 6 (SOS-Convex)**.** A polynomial $p(x)$ is SOS-convex if the following holds

$$u^T \nabla^2 p(x) u \in \sum[x, u] \tag{6}$$

where $u, x \in \mathbb{R}^n$. SOS-convexity is a sufficient condition for the Hessian of $p(x)$ to be positive semi-definite and therefore $p(x)$ to be convex.

In the development that follows, we will be interested in solving slight variations of the following problem.

$$\min_{P} \quad -\log\det P \tag{7a}$$

$$\text{s.t.}$$

$$P \succeq 0, \quad p(x) = z(x)^T P z(x), \tag{7b}$$

$$1 - p(x) \geq 0 \quad \forall\, x \in \mathcal{X}, \tag{7c}$$

Here $\mathcal{X}$ is a semi-algebraic set defined by n_i polynomial inequalities and n_j polynomial equalities.

$$\mathcal{X} = \{x \,|\, g_i(x) \geq 0, i \in [n_i], h_j(x) = 0, j \in [n_j]\} \tag{8}$$

Equation (7b) constrains $p(x)$ to be a SOS polynomial. Equation (7c) is a set-containment condition. The generalized $\mathcal{S}$-procedure provides a sufficient condition for the set-containment to hold [19]. For each polynomial equality $g_i(x)$ or inequality $h_j(x)$ describing the set $\mathcal{X}$, we introduce a non-negative polynomial $\lambda_i(x)$ or polynomial $\mu_j(x)$ respectively. The generalized $\mathcal{S}$-procedure involves replacing (7c) with the following:

$$1 - p(x) - \sum_i \lambda_i(x) g_i(x) - \sum_j \mu_j(x) h_j(x) \geq 0, \tag{9}$$

$$\lambda_i(x) \geq 0 \quad i \in [n_i] \tag{10}$$

By replacing the non-negativity constraints in (9), (10) with the more restrictive condition that the expressions be SOS polynomials, we obtain a semidefinite program which is readily solved.

$$\begin{aligned} &\min_{P,\, \lambda_{[1:n_i]}(x),\, \mu_{[1:n_j]}(x)} && -\log\det P \\ &\text{s.t.} \\ &P \succeq 0, \quad p(x) = z(x)^T P z(x), \\ &1 - p(x) - \sum_i \lambda_i(x) g_i(x) - \sum_j \mu_j(x) h_j(x) \in \sum[x], \\ &\lambda_i(x) \in \sum[x], \quad i \in [n_i]\,. \end{aligned} \tag{11}$$

Note when a polynomial is listed as a decision variable, e.g., $\lambda_{[1:n_i]}(x)$ and $\mu_{[1:n_j]}(x)$ underneath the $\min$, it is implied that the monomial basis is specified and the coefficients are decision variables.

Remark 1. Representing an equality constraint requires introducing a polynomial $\mu(x)$. In contrast, representing an inequality requires introducing a sum-of-squares polynomial multiplier $\lambda(x)$ which has a smaller feasible set and creates an additional semidefinite constraint. As such, it is generally advantageous to represent sets using equalities when applying the generalized $\mathcal{S}$-procedure.

In the development that follows we focus on transforming problems of interest into the form of (7). Once in this form, the subsequent application of the generalized $\mathcal{S}$-procedure is mechanical.

III. Problem Statement

We now setup the problem of optimization-based motion planning with collision avoidance constraints. For convenience, our notation closely follows that of [9].

A. Vehicle and Obstacle Models

Consider a vehicle with states $x_k \in \mathbb{R}^{n_x}$ and inputs $u_k \in \mathbb{R}^{n_u}$ at time step k. The dynamics evolve according to $x_{k+1} = f(x_k, u_k)$ where $f : \mathbb{R}^{n_x} \times \mathbb{R}^{n_u} \to \mathbb{R}^{n_x}$. The vehicle occupies space in $\mathbb{R}^n$. The vehicle's shape is assumed to be represented by n_b Euclidean balls with radii $r^{(i)}$.

$$\mathcal{B}^{(i)} = \{y \in \mathbb{R}^n |\ \|y\|_2 \leq r^{(i)}\},\ i \in [n_b]. \tag{12}$$

The center of each ball is a function of the vehicle's state as given by $t^{(i)} : \mathbb{R}^{n_x} \to \mathbb{R}^n$. Thus at time index k, the space occupied by ball i is given by:

$$\mathcal{V}^{(i)}(x_k) = \mathcal{B}^{(i)} \oplus t^{(i)}(x_k)\,. \tag{13}$$

The union $\bigcup_i \mathcal{V}^{(i)}(x_k)$ gives the total space occupied by the vehicle at time index k. For ease of exposition, in what follows we focus w.l.o.g. on the case when the vehicle is represented by a single ball ($n_b = 1$) and drop the superscript (i).

We assume there are M obstacles present in the environment indexed by $m \in [M]$. Each obstacle $\mathcal{O}^{(m)}$ is a polytope (closed, convex) with $k^{(m)}$ vertices $\{v_1, \ldots, v_{k^{(m)}}\}$ defining the convex hull as in (3). Equivalently represented in halfspace form (2), the obstacle m is defined by $j^{(m)}$ constraints given by $A^{(m)} \in \mathbb{R}^{j^{(m)} \times n}, b^{(m)} \in \mathbb{R}^{j^{(m)}}$.

B. Optimal Control Problem

We consider an optimal control problem of controlling the vehicle over N steps. The vehicle begins at start state x_S and must end at final state x_F. Let X, U denote the vector of all states and controls respectively, $X = [x_0^T, \ldots, x_N^T]^T, U = [u_0^T, \ldots, u_{N-1}]^T$. We seek to minimize an objective $l(X, U)$ where $l : X \times U \to \mathbb{R}$. Additionally, the vehicle is subject to n_h constraints given by $h(X, U) \leq 0$ where $h : X \times U \to \mathbb{R}^{n_h}$ and the inequality is interpreted element-wise. We assume that $l(X, U)$ and $h(X, U)$ are continuously differentiable and therefore suitable for nonlinear programming solvers which utilize gradient and Hessian information. Lastly, we enforce collision avoidance constraints between each obstacle and the vehicle. The resulting optimization problem takes the following form:

$$\min_{X, U} \quad l(X, U) \tag{14a}$$

$$\text{s.t.}$$

$$x_0 = x_S, \quad x_N = x_F, \tag{14b}$$

$$x_{k+1} = f(x_k, u_k), \quad k = 0, \ldots, N-1 \tag{14c}$$

$$h(X, U) \leq 0, \tag{14d}$$

$$\mathcal{V}(x_k) \cap \mathcal{O}^{(m)} = \emptyset, \quad k \in [N],\, m \in [M]. \tag{14e}$$

Equation (14e) represents the collision avoidance constraints which are non-convex and non-smooth in general. In [7], the authors provide an exact, smooth reformulation of these constraints. As the distance between two convex shapes can be computed using convex optimization, the authors leverage strong duality to develop necessary and sufficient conditions for a Euclidean ball of radius r to not intersect a given convex shape. This requires introducing dual variables associated with the halfspace constraints representing each obstacle

$\lambda_k^{(m)} \in \mathbb{R}^{j^{(m)}}$, $k \in [N]$, $m \in [M]$ and replacing (14e) with the following constraints.

$$\begin{aligned}(A^{(m)}t(x_k) - b^{(m)})^T\lambda_k^{(m)} &> r,\\ \|A^{(m)^T}\lambda_k^{(m)}\|_2^2 &\leq 1,\\ \lambda_k^{(m)} &\geq 0,\\ k \in [N],\, m \in [M].&\end{aligned} \tag{15}$$

If each obstacle has L halfspace constraints, this method introduces $(2+L)MN$ constraints and LMN dual variables which can result in a large nonlinear program that is computationally intensive. In the following, we present a method for approximating the collision avoidance constraints while introducing only MN constraints and no additional variables.

IV. Collision Avoidance via Minkowski Sums

We will utilize Minkowski sums to represent the collision avoidance constraints between a closed, convex polytope obstacle $\mathcal{O} = \{y \in \mathbb{R}^n \,|\, a_i^T y \leq b_i, i \in [L]\}$ and a vehicle with shape given by the Euclidean ball $\mathcal{B} = \{w \in \mathbb{R}^n | w^T w \leq r^2\}$. We first review a fundamental result from computational geometry.

Lemma 4. *Let $\mathcal{O}$ and $\mathcal{B}$ be sets in $\mathbb{R}^n$. Let $\mathcal{V} = \mathcal{B} \oplus t$ be the set $\mathcal{B}$ translated by $t \in \mathbb{R}^n$. Then the following relation holds:*

$$\mathcal{O} \cap \mathcal{V} \neq \emptyset \Leftrightarrow t \in \mathcal{O} \oplus (-\mathcal{B}) \tag{16}$$

Proof. See, e.g. [5], [20] □

In words, when $\mathcal{B}$ is located at position t, it makes contact with $\mathcal{O}$ if and only if t is in the Minkowski sum $\mathcal{O} \oplus -\mathcal{B}$. Thus collision avoidance with respect to obstacle $\mathcal{O}$ is equivalent to ensuring $t \notin \mathcal{O} \oplus -\mathcal{B}$.

When $\mathcal{O}$ is a polytope and $\mathcal{B}$ is a Euclidean ball, the set $\mathcal{O} \oplus (-\mathcal{B})$ is semi-algebraic. As such we cannot directly include the condition $t \notin \mathcal{O} \oplus (-\mathcal{B})$ as a constraint in a nonlinear optimization problem which requires closed-form, twice differentiable expressions.

Instead we propose to find an outer approximation $\mathcal{O} \oplus (-\mathcal{B}) \subseteq \widetilde{\mathcal{M}} \subset \mathbb{R}^n$ where $\widetilde{\mathcal{M}}$ is defined as the 1-level set of a function $p : \mathbb{R}^n \to \mathbb{R}$. Recall in our setting the translation of the ball at time index k is a function of the vehicle's state x_k as given by $t : \mathbb{R}^{n_x} \to \mathbb{R}^n$. Collision avoidance with respect to obstacle $\mathcal{O}$ can then be ensured by imposing the constraint $p(t(x_k)) > 1 \Leftrightarrow t(x_k) \notin \widetilde{\mathcal{M}} \Rightarrow t(x_k) \notin \mathcal{O} \oplus (-\mathcal{B}) \Leftrightarrow \mathcal{O} \cap \mathcal{V} = \emptyset$.

If multiple obstacles $\mathcal{O}^{(m)}, m \in [M]$ are present, we repeat this process for each obstacle and denote the associated function as $p^{(m)}(x)$. In our trajectory optimization problem we replace (14e) with MN constraints.

$$p(t(x_k))^{(m)} > 1, \quad k \in [N], \quad m \in [M]. \tag{17}$$

A. Outer Approximations of the Minkowski Sum

We would like our outer approximations to closely approximate the true set. To do so, we pose an optimization problem in which we minimize the volume of the outer approximation.

$$\begin{aligned}&\min_{p(x)} \quad \text{vol } \widetilde{\mathcal{M}}\\ &\text{s.t.}\\ &1 - p(x) \geq 0 \qquad \forall\, x \in \mathcal{O} \oplus (-\mathcal{B}),\\ &\widetilde{\mathcal{M}} = \{x \,|\, p(x) \leq 1\}\end{aligned} \tag{18}$$

In general we cannot solve this optimization problem. To arrive at a tractable formulation, we apply the generalized $\mathcal{S}$-procedure. We first parameterize the polynomial as $p(x) = z(x)^T P z(x)$ where $z(x)$ is a monomial basis chosen by the user and P is a positive semi-definite matrix of appropriate dimension. For arbitrary polynomials, we lack an expression for minimizing the volume of the 1-level set. Various heuristics have been proposed [14]–[16]. We have found maximizing the determinant of P, as proposed in [16], to work well for the problems herein. The resulting optimization problem is

$$\begin{aligned}&\min_{P} \quad -\log\det P\\ &\text{s.t.}\\ &p(x) = z(x)^T P z(x), \quad P \succeq 0,\\ &1 - p(x) \geq 0 \qquad \forall\, x \in \{y - w \,|\, a_i^T y \leq b_i,\\ &\qquad\qquad\qquad w^T w \leq r^2, i \in [L]\}\end{aligned} \tag{19}$$

where we have explicitly written the set resulting from the Minkowski sum in terms of y and w along with inequalities that ensure $y \in \mathcal{O}$ and $w \in \mathcal{B}$.

We apply the $\mathcal{S}$-procedure to replace the set-containment condition with a sufficient condition. This requires introducing multipliers $\lambda(y, w)$. We then replace the non-negativity conditions with the sufficient condition that the expression admits a sum-of-squares decomposition in terms of free variables y and w.

Optimization Problem 1: Outer Approximation

$$\begin{aligned}&\min_{P,\, \lambda_{[0:L]}(y,w)} \quad -\log\det P\\ &\text{s.t.}\\ &p(x) = z(x)^T P z(x), \quad P \succeq 0,\\ &1 - p(y-w) - \lambda_0(y,w)(r^2 - w^T w)\\ &\qquad - \sum_{i=1}^{L} \lambda_i(y,w)(b_i - a_i^T y) \in \textstyle\sum[y,w]\\ &\lambda_i(y,w) \in \textstyle\sum[y,w] \quad i = 0, \ldots, L\end{aligned} \tag{OA}$$

The formulation given by (OA) is viable but computationally expensive because the SOS decompositions involve both w and y giving $2n$ free variables for $x \in \mathbb{R}^n$. As we seek higher-order approximations, the monomial basis grows rapidly in size leading to large semidefinite programs. We

now develop a computationally cheaper program by leveraging convexity.

B. Convex Outer Approximations of the Minkowski Sum

In developing an efficient method for outer approximating the Minkowski sum, we will utilize the following Lemma.

Lemma 5. *Let $\mathcal{O} \subset \mathbb{R}^n$ be a polytope with K vertices $\{v_i\}, i \in [K]$. Let $\mathcal{B} \subset \mathbb{R}^n$ be a convex set that is the α-sublevel set of a function $f : \mathbb{R}^n \to \mathbb{R}$. Let S be any convex set in $\mathbb{R}^n$. Then the following criterion holds:*

$$\mathcal{O} \oplus (-\mathcal{B}) \subseteq S \Leftrightarrow \{v_i\} \oplus (-\partial\mathcal{B}) \subseteq S \tag{20}$$

Proof. First represent $\mathcal{O} \oplus (-\mathcal{B})$ in terms of its convex hull.

$$\begin{aligned} \mathcal{O} \oplus (-\mathcal{B}) &= \mathbf{conv}\{v_i\} \oplus \mathbf{conv}(-\partial\mathcal{B}), && \textbf{Lemma 1} \\ &= \mathbf{conv}[\{v_i\} \oplus (-\partial\mathcal{B})], && \textbf{Lemma 3} \end{aligned}$$

Next apply the property of the convex hull (1).

$$\mathbf{conv}[\{v_i\} \oplus (-\partial\mathcal{B})] \subseteq S \Leftrightarrow \{v_i\} \oplus (-\partial\mathcal{B}) \subseteq S$$

□

Lemma 5 provides a more efficient condition for outer approximating the Minkowski sum with a set $\widetilde{\mathcal{M}} = \{x \,|\, p(x) \leq 1\}$ as we only have to consider the vertices of $\mathcal{O}$ and the boundary of $\mathcal{B}$ in our set-containment constraint as follows.

$$1 - p(x) \geq 0 \quad \forall\, x \in v_i \oplus (-\partial\mathcal{B}) \quad i \in [K]. \tag{21}$$

However, it requires the condition that $\widetilde{\mathcal{M}}$ be a convex set. We argue that this is a reasonable constraint as $\mathcal{O} \oplus (-\mathcal{B})$ is itself convex. It is difficult to impose the condition that the 1-level set of $p(x)$, i.e., $\widetilde{\mathcal{M}}$, is convex. Recalling Definition 6, we will instead impose the sufficient condition that the function $p(x)$ be sos-convex.

As done previously, we rewrite the set-containment conditions using the generalized $\mathcal{S}$-procedure. We then replace the non-negativity conditions with SOS conditions.

Optimization Problem 2: Convex Outer Approximation

$$\begin{aligned} \min_{P,\, \mu_{[1:K]}(w)} \quad & -\log\det P \\ \text{s.t.} \quad & \\ & P \succeq 0,\, p(x) = z(x)^T P z(x) \qquad \text{(COA)} \\ & 1 - p(v_i - w) - \mu_i(w)(r^2 - w^T w) \in \textstyle\sum[w], i \in [K] \\ & u^T \nabla^2 p(x) u \in \textstyle\sum[x, u] \end{aligned}$$

Remark 2. The formulation of (COA) is advantageous in that the multipliers $\mu(w)$ do not have to be SOS and they only depend on n free variables ($w \in \mathbb{R}^n$). In contrast, (OA) requires SOS multipliers $\lambda(w, y)$ which depend on $2n$ free variables ($w, y \in \mathbb{R}^n$). The former leads to smaller semidefinite programs which scale better with respect to the dimension n or the complexity of $\mathcal{O}$. This is numerically illustrated in the following example.

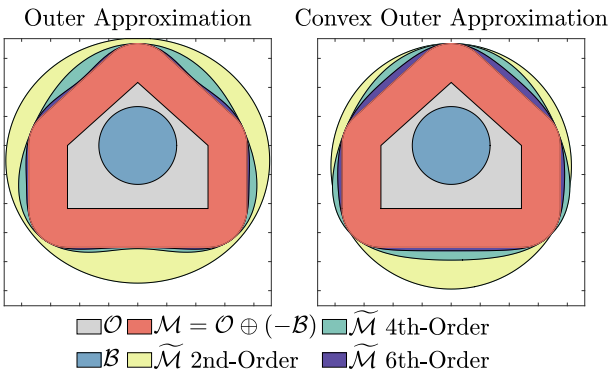

Fig. 2. Outer Approximation of Minkowski Sum

C. 2D Example

We generate 1000 random test cases in $\mathbb{R}^2$. For each case we generate a polytope $\mathcal{O}$ with $n \in \{3, 4, \ldots, 12\}$ vertices $v_i \in [-1, 1]^2, i = 1, \ldots, n$ along with a disk $\mathcal{B}$ with radius $r \in [0, 1]$. We form outer approximations $\widetilde{\mathcal{M}} = \{x \,|\, p(x) \leq 1\}$ of the set $\mathcal{M} = \mathcal{O} \oplus (-\mathcal{B})$ using both (OA) and (COA). For each we consider polynomials $p(x)$ of degree 2, 4 and 6. To assess the accuracy of our outer approximations, we compute the approximation error as $\frac{Area(\widetilde{\mathcal{M}})}{Area(\mathcal{M})} - 1$. Table I lists the mean approximation error of the 1000 test cases. Empirically, as we increase the polynomial order, the approximation error is reduced, indicating we are getting better outer approximations. Table II lists the mean solve times. As expected, (COA) has significantly faster solve times than (OA) due to the smaller semidefinite program. Figure 2 provides an example of the results.

Polynomial Degree	2	4	6
Outer Approximation	40%	9%	3%
Convex Outer Approximation	25%	9%	5%

TABLE I

MEAN APPROXIMATION ERROR OF MINKOWSKI SUMS

Polynomial Degree	2	4	6
Outer Approximation	0.020	0.174	0.925
Convex Outer Approximation	0.004	0.014	0.049

TABLE II

MEAN SOLVE TIMES (S) OF OPTIMIZATION PROBLEMS 1 & 2

Remark. For the case when $p(x)$ is a quadratic, the resulting minimum volume $\widetilde{\mathcal{M}}$ can be found exactly using the semidefinite program for finding the minimum volume outer ellipsoid (MVOE) covering a union of ellipsoids [6]. In this case each ellipsoid is a ball of radius r centered at vertex v_i. Our convex formulation (COA) can be seen as a generalized form of this result. The non-convex case (OA) has a smaller feasible set due to the reliance on SOS multipliers and does not return the minimum volume outer ellipsoid in general. Thus (OA) is only advantageous when seeking non-ellipsoidal approximations.

V. Motion Planning Examples

A. Autonomous Car

We demonstrate our proposed obstacle avoidance conditions on a motion planning problem for an autonomous car. We adopt the autonomous racing car model from [3]. The model has 6 states, $x = \begin{bmatrix} p_x & p_y & \psi & v_x & v_y & \omega \end{bmatrix}^T$ consisting of position (p_x, p_y), orientation (ψ), body velocities (v_x, v_y) and yaw rate (ω). The inputs are motor duty cycle (d) and steering angle (δ). Due to space constraints we refer the reader to [3] for more details of the dynamic model. We represent the vehicle's shape as a single disk $\mathcal{B}$ of radius $r = 0.067$m. The center of the disk at time step k is simply the $(p_{x,k}, p_{y,k})$ position of the vehicle: $t(x_k) = \begin{bmatrix} p_{x,k} & p_{y,k} \end{bmatrix}^T$.

We consider a situation in which the vehicle is making forward progress along a straight track while navigating obstacles. We pose this as a trajectory optimization problem of the form (14). The objective is to minimize the 2-norm of the input, $l(X, U) = \|U\|_2^2$. The vehicle starts at state $x_S = \begin{bmatrix} 0 & p_{y,S} & 0 & 1 & 0 & 0 \end{bmatrix}^T$ and must end at state $x_F = \begin{bmatrix} 3 & p_{y,F} & 0 & 1 & 0 & 0 \end{bmatrix}^T$. The dynamic constraints (14c) are implemented using a 4th-order Runge-Kutta integration of the dynamics. We use a time-step of $0.03s$ over $N = 100$ steps, giving a 3s time horizon. At each step $k = 0, \ldots, N-1$, the vehicle is subject to box constraints on the position $p_{y,k} \in [0, 0.3]$ and inputs $d_k \in [-0.1, 1], \delta_k \in [-1, 1]$.

We consider scenarios consisting of $M \in \{1, 2, \ldots, 10\}$ obstacles. For each scenario, we generate 100 random test cases in which we vary the start and final y position, $p_{y,S}, p_{y,F} \in [0, 0.3]$ along with the placement and shapes of the M obstacles. We solve (14)using both the exact representation (15) of the collision avoidance constraints and the approximate representation (17). We use 4th-order, convex polynomials to represent the Minkowski sum approximations.

B. Initialization

In optimization-based motion planning around obstacles, it is advantageous to supply a collision-free initial path to the solver. Similar to [7], we use $A^\star$ [21], [22] to find a minimum-distance collision free path on a discretized representation of the environment. This path does not consider the dynamics and is generally not kinematically feasible. We use this to initialize our guess for the vehicle's position over time. The same initialization is used for both the exact and approximate methods. For the exact method, similar to [7], we initialize the dual variables λ to 0.05.

C. Results

Figure 3 shows a scenario in which the vehicle navigates six obstacles. We plot the obstacles in grey along with the exact and approximate Minkowski sums of each obstacle and the vehicle $\mathcal{B}$. As the exact method is equivalent to ensuring the vehicle's position (p_x, p_y) remains outside the exact Minkowski sums, this helps to visualize the conservatism of our outer approximations. In this case, the 4th-order approximate representations are quite tight and are only visible as thin green borders around the exact Minkowski sums in red. For reference, we also plot 2nd-order, ellipsoidal approximations which are unacceptably conservative and do not admit a collision-free path. As the objective penalizes large steering and acceleration commands, the vehicle naturally makes tight maneuvers around the obstacles. The exact method achieves a slightly lower cost because the configuration-space obstacles it must avoid are slightly smaller, requiring less maneuvering. However the difference is negligible and the resulting trajectories are nearly identical.

Figure 4 shows the solve time statistics of the approximate and exact methods as we vary the number of obstacles present. Both methods exhibit linear trends in median solve time as the number of obstacles increases. The approximate method solves 1.7x faster than the exact method when just one obstacle is present. With ten obstacles present, the approximate method solves 4.0x faster. The approximate method shows less variability in the solve times, with a maximum solve time of 0.25s. For the exact method, 9 of the 1000 cases either did not converge or exceeded the maximum allowed solve time of 1.5s. To evaluate the impact of using the approximate method, we normalize the cost of the trajectories by the cost of the trajectories obtained with the exact method, $1 - \frac{J_{approx}}{J_{exact}}$. For 851 of the 991 cases in which the exact method was successfully solved, the approximate method returned a trajectory less than 0.1% sub-optimal. The worst-case sub-optimality was 1.5%.

Remark 3. The solver times reported for the approximate method only reflect the time spent solving the nonlinear program. We do not include the time required to compute the outer approximations. In a real-time motion planning problem, these approximations would only be performed once per obstacle, either offline or online. We note that based on Table II, approximating an obstacle with a 4th-order convex polynomial takes 0.014s. If this computation time were included in Figure 4, the approximate method would still be consistently faster than the exact method.

D. Quadcopter

Next we demonstrate our obstacle avoidance conditions on a 3D motion planning problem for a quadcopter model taken from [23]. The model has 12 states, $x = \begin{bmatrix} p_x & p_y & p_z & \phi & \theta & \psi & v_x & v_y & v_z & p & q & r \end{bmatrix}^T$ consisting of position (p_x, p_y, p_z) and velocity (v_x, v_y, v_z) of the quadcopter's center-of-gravity in a world reference frame, Euler angles (ϕ, θ, ψ), and body rates (p, q, r). The inputs are the four rotor speeds $(\omega_i, i = 1, \ldots, 4)$. Due to space constraints we refer the reader to [7], [23] for more details of the dynamic model.

We represent the quadcopter's shape as a single ball $\mathcal{B}$ of radius $r = 0.25$m. The center of the disk at time step k is simply the position of the quadcopter: $t(x_k) = \begin{bmatrix} p_{x,k} & p_{y,k} & p_{z,k} \end{bmatrix}^T$.

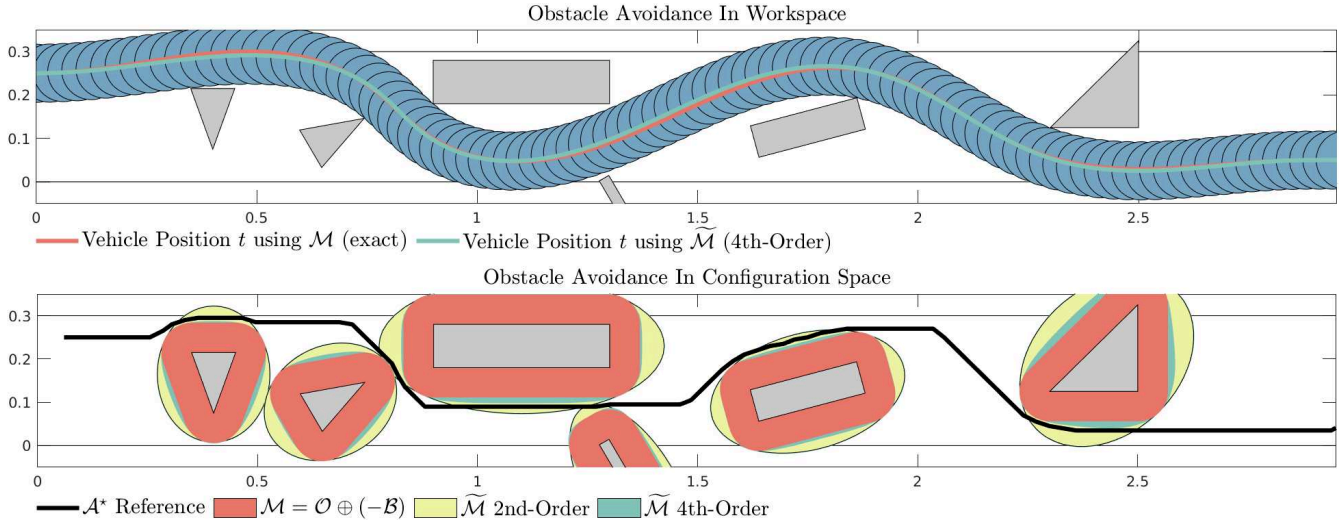

Fig. 3. Autonomous Car Navigating Obstacles In Workspace (Upper) and Configuration Space (Lower)

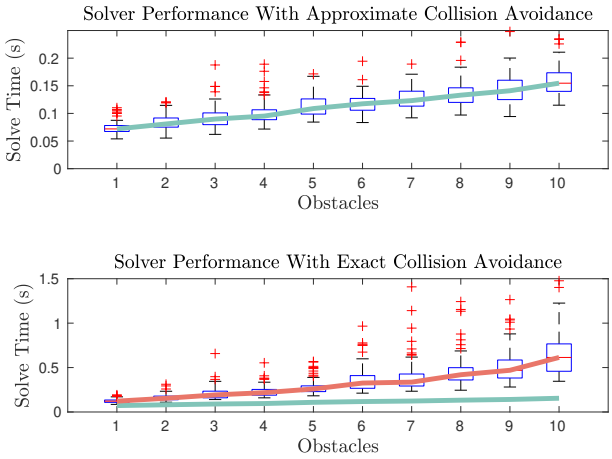

Fig. 4. Solve Time Statistics for Autonomous Car Example

	Min. (s)	Median (s)	Max. (s)
Approx. Collision Avoidance	0.371	0.672	3.432
Exact Collision Avoidance	3.053	7.379	22.070

TABLE III

SOLVE TIMES STATISTICS FOR QUADCOPTER EXAMPLE

We consider a situation in which the quadcopter is navigating a cluttered room of dimensions $10 \times 10 \times 5$. The quadcopter starts at the origin with state $x_S = \mathbf{0}_{12}$ and must end at state $x_F = \begin{bmatrix} p_{x,F} & p_{y,F} & p_{z,F} & \mathbf{0}_9 \end{bmatrix}^T$ while avoiding any obstacles. Here $\mathbf{0}_i$ denotes the zero vector in $\mathbb{R}^i$. We pose this as a trajectory optimization problem of the form (14). The objective is to minimize the 2-norm of the input, $l(X,U) = \|U\|_2^2$. The dynamic constraints (14c) are implemented using a 4th-order Runge-Kutta integration of the dynamics. We use a time-step of $0.04s$ over $N = 100$ steps, giving a 4s time horizon. At each step $k = 0, \ldots, N-1$, the vehicle is subject to box constraints on the position $p_{x,k} \in [0,10], p_{y,k} \in [0,10], p_{z,k} \in [0,5]$ and inputs $\omega_{i,k} \in [1.2, 7.8], i = 1, \ldots, 4$.

The environment contains 20 obstacles. We consider 92 different final positions. For each we generate a collision-free initial guess using $A^\star$. We solve (14) using both the exact representation (15) of the collision avoidance constraints and the approximate representation (17). For the approximate case we use 4th-order, convex polynomials to represent the Minkowski sums. Table III lists the resulting solve times of the nonlinear program. Of the 92 test cases, the approximate method was less than 0.1% sub-optimal for 66 of the 92 test cases. The worst-case was 1.5% sub-optimal. The resulting trajectories of the exact and approximate methods were nearly identical but the approximate method solved an order-of-magnitude faster. The upper plot of Figure 5 shows the quadcopter navigating the cluttered environment. The lower plot gives the configuration-space view with the Minkowksi sum approximations shown in red.

E. Implementation Details

All examples were solved on a MacBook Pro with a 2.6 GHz 6-Core Intel Core i7 CPU. YALMIP [24] was used in conjunction with MOSEK [25] to solve the sum-of-squares optimization problems. IPOPT [26] with the MA97 linear solver [27] was used to solve the nonlinear optimization problems with exact gradients and Hessians supplied by CasADi [28].

VI. CONCLUSIONS

Optimization-based approaches to trajectory generation for robots, autonomous vehicles, and quadcopters are appealing for their great flexibility in the types of dynamics, objectives, and constraints they can accommodate. One limitation of these methods is that it can be difficult to efficiently represent obstacle avoidance conditions in a manner acceptable to the solvers. This work presented novel obstacle avoidance

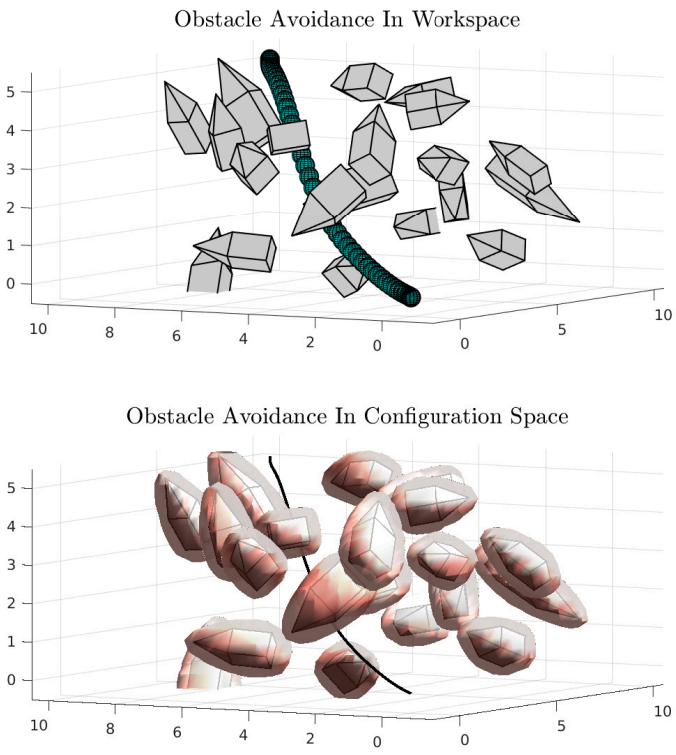

Fig. 5. Quadcopter Navigating Obstacles In Workspace (Upper) and Configuration Space (Lower)

conditions based on outer approximations of Minkowski sums. This method is advantageous in that yields a much smaller nonlinear program compared to exactly representing the collision avoidance conditions. On motion planning problems for an autonomous vehicle and quadcopter, the approximate method solved 4x and 10x faster respectively when navigating cluttered environments. The resulting trajectories exhibited negligible sub-optimality compared to using exact collision avoidance conditions. Currently our method is limited to cases in which the vehicle is represented by a union of Euclidean balls and the obstacle is a bounded, convex polytope. In future work we plan to consider more complicated vehicle shapes and non-convex obstacles.

REFERENCES

[1] H. Bock and K. Plitt, "A multiple shooting algorithm for direct solution of optimal control problems*," *IFAC Proceedings Volumes*, vol. 17, no. 2, pp. 1603–1608, 1984. 9th IFAC World Congress: A Bridge Between Control Science and Technology, Budapest, Hungary, 2-6 July 1984.

[2] L. T. Biegler, "Solution of dynamic optimization problems by successive quadratic programming and orthogonal collocation," *Computers & Chemical Engineering*, vol. 8, pp. 243–247, 1984.

[3] A. Liniger, A. Domahidi, and M. Morari, "Optimization-based autonomous racing of 1:43 scale rc cars," *Optimal Control Applications and Methods*, vol. 36, no. 5, pp. 628–647, 2015.

[4] M. Brown and J. Gerdes, "Coordinating tire forces to avoid obstacles using nonlinear model predictive control," *IEEE Transactions on Intelligent Vehicles*, vol. PP, pp. 1–1, 11 2019.

[5] S. M. LaValle, *Planning Algorithms*. Cambridge University Press, 2006.

[6] S. Boyd and L. Vandenberghe, *Convex Optimization*. Cambridge University Press, March 2004.

[7] X. Zhang, A. Liniger, and F. Borrelli, "Optimization-based collision avoidance," *IEEE Transactions on Control Systems Technology*, vol. 29, no. 3, pp. 972–983, 2021.

[8] R. B. Patel and P. J. Goulart, "Trajectory generation for aircraft avoidance maneuvers using online optimization," *Journal of Guidance, Control, and Dynamics*, vol. 34, no. 1, pp. 218–230, 2011.

[9] X. Zhang, A. Liniger, A. Sakai, and F. Borrelli, "Autonomous parking using optimization-based collision avoidance," in *2018 IEEE Conference on Decision and Control (CDC)*, pp. 4327–4332, 2018.

[10] S. Ruan and G. S. Chirikjian, "Closed-form minkowski sums of convex bodies with smooth positively curved boundaries," 2020.

[11] Y. Yan and G. S. Chirikjian, "Closed-form characterization of the minkowski sum and difference of two ellipsoids," *Geometriae Dedicata*, vol. 177, no. 1, pp. 103–128, 2015.

[12] S. Ruan, K. L. Poblete, Y. Li, Q. Lin, Q. Ma, and G. S. Chirikjian, "Efficient exact collision detection between ellipsoids and superquadrics via closed-form minkowski sums," in *2019 International Conference on Robotics and Automation (ICRA)*, pp. 1765–1771, 2019.

[13] C. M. Hoffmann, "Conversion methods between parametric and implicit curves and surfaces," 1990.

[14] A. Magnani, S. Lall, and S. Boyd, "Tractable fitting with convex polynomials via sum-of-squares," in *Proceedings of the 44th IEEE Conference on Decision and Control*, pp. 1672–1677, 2005.

[15] F. Dabbene, D. Henrion, and C. Lagoa, "Simple approximations of semialgebraic sets and their applications to control," *Automatica*, vol. 78, 09 2015.

[16] A. A. Ahmadi, G. Hall, A. Makadia, and V. Sindhwani, "Geometry of 3d environments and sum of squares polynomials," in *Robotics: Science and Systems*, 2017.

[17] J. Guthrie and E. Mallada, "Outer approximations of minkowski operations on complex sets via sum-of-squares optimization," in *2021 American Control Conference (ACC)*, pp. 2367–2373, 2021.

[18] R. Schneider, *Convex Bodies: The Brunn–Minkowski Theory*. Encyclopedia of Mathematics and its Applications, Cambridge University Press, 2 ed., 2013.

[19] P. Parrilo, *Structured semidefinite programs and semialgebraic geometry methods in robustness and optimization*. PhD thesis, California Institute of Technology, 2000.

[20] M. Berg, de, O. Cheong, M. Kreveld, van, and M. Overmars, *Computational geometry : algorithms and applications*. Germany: Springer, 3rd ed., 2008.

[21] P. E. Hart, N. J. Nilsson, and B. Raphael, "A formal basis for the heuristic determination of minimum cost paths," *IEEE Transactions on Systems Science and Cybernetics*, vol. 4, no. 2, pp. 100–107, 1968.

[22] P. E. Hart, N. J. Nilsson, and B. Raphael, "Correction to "a formal basis for the heuristic determination of minimum cost paths"," *SIGART Bull.*, p. 28–29, Dec. 1972.

[23] D. Mellinger, N. Michael, and V. Kumar, "Trajectory generation and control for precise aggressive maneuvers with quadrotors," *The International Journal of Robotics Research*, vol. 31, no. 5, pp. 664–674, 2012.

[24] J. Lofberg, "Yalmip : a toolbox for modeling and optimization in matlab," in *2004 IEEE International Conference on Robotics and Automation (IEEE Cat. No.04CH37508)*, pp. 284–289, Sep. 2004.

[25] M. ApS, *The MOSEK optimization toolbox for MATLAB manual. Version 8.1.*, 2017.

[26] 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.

[27] J. Hogg and J. A. Scott, "Hsl_ma97 : a bit-compatible multifrontal code for sparse symmetric systems," *Rutherford Appleton Laboratory Technical Reports*, 2011.

[28] J. A. E. Andersson, J. Gillis, G. Horn, J. B. Rawlings, and M. Diehl, "CasADi – A software framework for nonlinear optimization and optimal control," *Mathematical Programming Computation*, vol. 11, no. 1, pp. 1–36, 2019.