A Definition of Archimedean

In this section we formally define the Archimedean property that appears in Theorem 1.

Definition 1. *A semialgebraic set* $\mathcal{S}_g = \{x \mid g_i(x) \geq 0, i \in [n]\}$ *is Archimedean if there exists* $N \in \mathbb{N}$ *and* $\lambda_i(x) \in \boldsymbol{\Sigma}$ *such that:*

$$N - \sum_{i=1}^{n} x_i^2 = \lambda_0(x) + \sum_{i=1}^{n} \lambda_i(x) g_i(x)$$

B Practical Aspects

In this section we discuss important ways to scale Algorithm 1. In sections B.1 and B.2 we describe design choices which dramatically reduce the size of the SOS programs used in the alternations. Next, we discuss aspects of Algorithm 1 which can be parallelized, reducing solve time in the alternations. Finally, we describe an extension to the original Iris algorithm [1] which can be used to rapidly propose a very large region that is likely, but not certified, to be collision-free. Seeding Algorithm 1 with such a region can dramatically reduce the number of iterations required to obtain a satisfyingly large volume.

B.1 Choosing the Reference Frame

The polynomial implications upon which the programs (11) and (12) are based require choosing a coordinate frame between each collision pair $\mathcal{A}$ and $\mathcal{B}$. However, as the collision-free certificate between two different collision pairs can be computed independently of each other, we are free to choose a different coordinate frame to express the kinematics for each collision pair. This is important in light of (4) and (5) that indicate that the degree of the polynomials ${}^{F}f^{\mathcal{A}_j}$ and ${}^{F}g^{\mathcal{A}_j}$ are equal to two times the number of joints lying on the kinematic chain between frame F and the frame for $\mathcal{A}$. For example, the tangent-configuration space polynomial in the variable s describing the position of the end-effector of a 7-DOF robot is of total degree 14 when written in the coordinate frame of the robot base. However, when written in the frame of the third link, the polynomial describing the position of the end effector is only of total degree $(7-3) \times 2 = 8$. This observation is also used in [2] to reduce the size of the optimization program.

The size of the semidefinite variables in (11) and (12) scale as the square of the degree of the polynomial used to express the forward kinematics. Supposing there are n links in the kinematics chain between $\mathcal{A}$ and $\mathcal{B}$, then choosing the jth link along the kinematics chain as the reference frame F leads to scaling of order $j^2 + (n-j)^2$. Choosing the reference frame in the middle of the chain minimizes this complexity to scaling of order $\frac{n^2}{2}$ and we therefore adopt this convention in our experiments.

B.2 Basis Selection

The condition that a polynomial can be written as a sum of squares can be equivalently formulated as an equality constraint between the coefficients of the polynomial and an associated semidefinite variable known as the Gram matrix [3]. In general, a polynomial in k variables of total degree $2d$ has $\binom{k+2d}{2d}$ coefficients and requires a Gram matrix of size $\binom{k+d}{d}$ to represent which can quickly become prohibitively large. Fortunately, the polynomials in our programs contain substantially more structure which will allow us to drastically reduce the size of the Gram matrices.

We begin by noting from (6) that while both the numerator and denominator of the forward kinematics are of total degree $2n$, with n the number of links of the kinematics chain between frame A and F, both polynomials are of *coordinate* degree of at most two (i.e. the highest degree of s_i in any term is s_i^2). We will refer to this basis as $\mu(s)$ which is a vector containing terms of the form $\prod_{i=1}^{n} s_i^{d_i}$ with $d_i \in \{0, 1, 2\}$ for all n^3 possible permutations of the exponents d_i.

Next, we recall the form of $\alpha^{F,\mathcal{A}_j}(a_{\mathcal{A},\mathcal{B}}, b_{\mathcal{A},\mathcal{B}}, s)$ from (8b) and (8c). If $a_{\mathcal{A},\mathcal{B}}(s) = a_{\mathcal{A},\mathcal{B}}^T \eta(s)$ and $b_{\mathcal{A},\mathcal{B}}(s) = b_{\mathcal{A},\mathcal{B}}^T \eta(s)$ for some basis η in the variable s, then $\alpha^{F,\mathcal{A}_j}$ and $\beta^{F,\mathcal{A}_j}$ can be expressed as linear functions of the basis $\gamma(s) = \mathbf{vect}(\eta(s)\mu(s)^T)$ where we use **vect** to indicate the flattening of the matrix outer product. Concretely, if we choose to make $a_{\mathcal{A},\mathcal{B}}(s)$ and $b_{\mathcal{A},\mathcal{B}}(s)$ linear functions of the indeterminates s, then $\eta(s) = l(s) = \begin{bmatrix}1 & s_1 & \dots & s_n\end{bmatrix}$. Therefore $\alpha^{F,\mathcal{A}_j}$ and $\beta^{F,\mathcal{A}_j}$ can be expressed as linear functions of the basis

$$\gamma(s) = \begin{bmatrix}\mu(s) & s_1\mu(s) & \dots & s_n\mu(s)\end{bmatrix} \tag{13}$$

After choosing the basis $\eta(s)$, which determines the basis $\gamma(s)$, the equality constraints (9a) and (9b) constrain the necessary basis needed to express the multiplier polynomials $\lambda(s)$ and $\nu(s)$. The minimal such basis is related to an object known in computational algebra as the Newton polytope of a polynomial $\mathbf{New}(f)$ [4]. The exact condition is that the $\mathbf{New}(\eta(s)) + \mathbf{New}(\mu(s)) \subseteq \mathbf{New}(\rho(s)) + \mathbf{New}(l(s))$ where the sum in this case is the Minkowski sum.

If we choose $\eta(s)$ as the linear basis $l(s)$, then we obtain the condition that $\mathbf{New}(\rho(s)) = \mathbf{New}(\mu(s))$ and since $\mu(s)$ is a dense, even degree basis then we must take $\rho(s) = \mu(s)$. Choosing $\eta(s)$ as the constant basis would in fact result in the same condition, and therefore searching for separating planes which are linear functions of the tangent-configuration space does not increase the size of the semidefinite variables. As the complexity of (11) and (12) is dominated by the size of these semidefinite variables, separating planes which are linear functions changes does not substantially affect the solve time but can dramatically increase the size of the regions which we can certify.

Remark 3. In the case of certifying that the end-effector of a 7-DOF robot will not collide with the base using linearly parametrized hyperplanes, choosing to express conditions (9a) and (9b) in the world frame with naïvely chosen bases would result in semidefinite variables of size $\binom{7+7}{7} = 3432$. Choosing to express the conditions according to the discussion in Section B.1 and choosing the basis $\gamma(s)$ from (13) results in semidefinite matrices of rows at most $2^4 = 16$.

B.3 Parallelization

While it is attractive from a theoretical standpoint to write (11) as a single, large program it is worth noting that can in fact be viewed as $K+1$ individual SOS and SDP programs, where K is the number of collision pairs in the environment. Indeed, certifying whether pairs $(\mathcal{A}_1, \mathcal{A}_2)$ are collision-free for all s in the polytope $\mathcal{P}$ can be done completely independently of the certification of another pair $(\mathcal{A}_1, \mathcal{A}_3)$ as neither the constraints nor the cost couple the conditions of imposed on any pairs. Similarly, the search for the largest inscribed ellipsoid can be done independently of the search for the separating hyperplanes.

Solving the certification problem embedded in (11) as K individual SOS programs has several advantages. First, as written (11) has $2(m+1)K\sum_i |\mathcal{A}_i|$ semidefinite variables of various sizes. In the example from Section 5.1 this corresponds to 35,072 semidefinite variables. This can be prohibitively large to store in memory as a single program. Additionally, solving the problems independently enables us to determine which collision bodies cannot be certified as collision-free and allows us to terminate our search as soon as a single pair cannot be certified. Finally, decomposing the problems into subproblems enables us to increase computation speed by leveraging parallel processing.

We note that (12) cannot be similarly decomposed as on this step the variables c_i^T and d affect all of the constraints. However, this program is substantially smaller as we have fixed $2mK\sum_i |\mathcal{A}_i|$ semidefinite variables as constants and replaced them with $2m$ linear variables representing the polytope. This program is much more amenable to being solved as a single program.

B.4 Seeding the Algorithm

It is worth noting that the alternations in Algorithm 1 must be initialized with a polytope $\mathcal{P}_0$ for which (11) is feasible. In principle, the alternation proposed in Section 4.3 can be seeded with an arbitrarily small polytope around a collision-free seed point. This seed polytope is then allowed to grow using Algorithm 1. However, this may require running several dozens of iterations of Algorithm 1 for each seed point which can become prohibitive as the size of the problem grows. It is therefore advantageous to seed with as large a region as can be initially certified.

Here we discuss an extension of the IRIS algorithm in [1] which uses nonlinear optimization to rapidly generate large regions in C-space. These regions are not guaranteed to be collision-free and therefore they must still be passed to Algorithm 1 to be certified, but do provide good initial guesses. In this section, we will assume that the reader is familiar with IRIS and will only discuss the modification required to use it to grow C-space regions. Detailed pseudocode is available in Appendix C

IRIS grows regions in a given space by alternating between two subproblems: SEPARATINGHYPERPLANES and INSCRIBEDELLIPSOID. The INSCRIBEDELLIPSOID is exactly the program described in [5, Section 8.4.2] and we do not need

to modify it. The subproblem SeparatingHyperplanes finds a set of hyperplanes which separate the ellipse generated by InscribedEllipsoid from all of the obstacles. This subproblem is solved by calling two subroutines ClosestPointOnObstacle and TangentPlane. The former finds the closest point on a given obstacle to the ellipse, while the latter places a plane at the point found in ClosestPointOnObstacle that is tangent to the ellipsoid.

The original work in [1] assumes convex obstacles which enables ClosestPointOnObstacle to be solved as a convex program and for the output of TangentPlane to be globally separating plane between the obstacle and the ellipsoid of the previous step. Due to the non-convexity of the C-space obstacles in our problem formulation, finding the closest point on an obstacle exactly becomes a computationally difficult problem to solve exactly [6]. Additionally, placing a tangent plane at the nearest point will be only a locally separating plane, not a globally separating one.

To address the former difficulty, we formulate ClosestPointOnObstacle as a nonlinear program. Let the current ellipse be given as $\mathcal{E} = \{Qu + s_0 \mid \|u\|_2 \leq 1\}$ and suppose we have the constraint that $s \in \mathcal{P} = \{s \mid Cs \leq d\}$. Let $\mathcal{A}$ and $\mathcal{B}$ be two collision pairs and ${}^{\mathcal{A}}p_{\mathcal{A}}, {}^{\mathcal{B}}p_{\mathcal{B}}$ be some point in bodies $\mathcal{A}$ and $\mathcal{B}$ expressed in some frame attached to $\mathcal{A}$ and $\mathcal{B}$. Also, let ${}^{W}X^{\mathcal{A}}(s)$ and ${}^{W}X^{\mathcal{B}}(s)$ denote the rigid transforms from the reference frames $\mathcal{A}$ and $\mathcal{B}$ to the world frame respectively. We remind the reader that this notation is drawn from [7]. The closest point on the obstacle subject to being contained in $\mathcal{P}$ can be found by solving the program

$$\min_{s, {}^{\mathcal{A}}p_{\mathcal{A}}, {}^{\mathcal{B}}p_{\mathcal{B}}} (s - s_0)^T Q^T Q (s - s_0) \text{ subject to} \tag{14a}$$

$${}^{W}X^{\mathcal{A}}(s) {}^{\mathcal{A}}p_{\mathcal{A}} = {}^{W}X^{\mathcal{B}}(s) {}^{\mathcal{B}}p_{\mathcal{B}} \tag{14b}$$

$$Cs \leq d \tag{14c}$$

This program searches for the nearest configuration in the metric of the ellipse such that two points in the collision pair come into contact. We find a locally optimal solution $(s^\star, {}^{\mathcal{A}}p^\star_{\mathcal{A}}, {}^{\mathcal{B}}p^\star_{\mathcal{B}})$ to the program using a fast, general-purpose nonlinear solver such as SNOPT [8]. The tangent plane to the ellipse $\mathcal{E}$ at the point $s^\star$ is computed by calling TangentPlane then appended to the inequalities of $\mathcal{P}$ to form $\mathcal{P}'$. This routine is looped until (14) is infeasible at which point InscribedEllipse is called again.

Once a region $\mathcal{P} = \{s \mid Cs \leq d\}$ is found by Algorithm 2, it will typically contain some minor violations of the non-collision constraint. To find an initial, feasible polytope $\mathcal{P}_0$ to use in Algorithm 1, we search for a minimal uniform contraction δ of $\mathcal{P}$ such that $\mathcal{P}_\delta = \{s \mid Cs \leq d - \delta * 1\}$ is collision-free. This can be found by bisecting over the variable $\delta \in [0, \delta_{\max}]$ and solving repeated instances of (11).

Seeding Algorithm 1 with a $\mathcal{P}_0$ as above can dramatically reduce the number of alternations required to obtain a fairly large region and is frequently faster than seeding Algorithm 1 with an arbitrarily small polytope.

C Supplementary Algorithms

We present a pseudocode for the algorithm presented in Appendix B.4. A mature implementation of this algorithm can be found in Drake[7].

Algorithm 2: Given an initial tangent-configuration space point s_0 and a list of obstacles $\mathcal{O}$, return a polytopic region $\mathcal{P} = \{s \mid Cs \leq d\}$ and inscribed ellipsoid $\mathcal{E}_\mathcal{P} = \{s \mid Qu + s_c\}$ which contains a substantial portion of the free C-space (but is not guaranteed to contain no collisions)

$(C, d) \leftarrow$ plant joint limits
$\mathcal{P}_i \leftarrow \{s \mid Cs \leq d\}$
$\mathcal{E}_{\mathcal{P}_0} \leftarrow$ InscribedEllipsoid$(\mathcal{P}_0)$
$j \leftarrow$ number of rows of C
do
  do
    $(s^\star, {}^\mathcal{A}p^\star_\mathcal{A}, {}^\mathcal{B}p^\star_\mathcal{B}) \leftarrow$ FindClosestCollision$(\mathcal{P}_i, \mathcal{E}_{\mathcal{P}_i})$
    $(c^T_{j+1}, d_{j+1}) \leftarrow$ TangentHyperplane$(s^\star, \mathcal{E}_{\mathcal{P}_i})$
    $C \leftarrow$ **hstack**(C, c^T_{j+1})
    $d \leftarrow$ **hstack**(d, d_{j+1})
    $\mathcal{P}_i \leftarrow \{s \mid Cs \leq d\}$
    $j \leftarrow j + 1$
  while FindClosestCollision$(\mathcal{P}_i, \mathcal{E}_{\mathcal{P}_i})$ *is feasible*;
  $\mathcal{E}_{\mathcal{P}_i} \leftarrow$ InscribedEllipsoid$(\mathcal{P}_i)$ $i \leftarrow i + 1$
while $(\boldsymbol{vol}(\mathcal{E}_i) - \boldsymbol{vol}(\mathcal{E}_{i-1}))\,/\,\boldsymbol{vol}(\mathcal{E}_{i-1}) \geq$ *tolerance*;
return $(\mathcal{P}_i, \mathcal{E}_{\mathcal{P}_i})$

References

1. R. Deits and R. Tedrake, "Computing large convex regions of obstacle-free space through semidefinite programming," in *Algorithmic foundations of robotics XI*. Springer, 2015, pp. 109–124.
2. P. Trutman, S. E. D. Mohab, D. Henrion, and T. Pajdla, "Globally optimal solution to inverse kinematics of 7dof serial manipulator," *arXiv preprint arXiv:2007.12550*, 2020.
3. P. A. Parrilo, "Sum of squares programs and polynomial inequalities," in *SIAG/OPT Views-and-News: A Forum for the SIAM Activity Group on Optimization*, vol. 15, no. 2.
4. B. Sturmfels, "On the newton polytope of the resultant," *Journal of Algebraic Combinatorics*, vol. 3, no. 2.

[7] https://github.com/RobotLocomotion/drake/blob/2f75971b66ca59dc2c1dee4acd78952474936a79/geometry/optimization/iris.cc#L440

5. S. Boyd, S. P. Boyd, and L. Vandenberghe, *Convex optimization*. Cambridge university press, 2004.
6. C. Ferrier, "Computation of the distance to semi-algebraic sets," *ESAIM: Control, Optimisation and Calculus of Variations*, vol. 5.
7. R. Tedrake, *Robotic Manipulation*, 2021. [Online]. Available: https://manipulation.mit.edu/pick.html#monogram
8. P. E. Gill, W. Murray, and M. A. Saunders, "Snopt: An sqp algorithm for large-scale constrained optimization," *SIAM review*, vol. 47, no. 1.