Intersection of Box and Finite Cylinder

David Eberly, Geometric Tools, Redmond WA 98052
https://www.geometrictools.com/

Created: December 21, 2021

Contents

1 Introduction

An intersection algorithm is called a *test-intersection* query when the outcome is simply whether the two objects intersect. The algorithm is a *find-intersection* query when the outcome is whether the two objects intersect and includes information about the set of intersection (if it is nonempty). This document describes test-intersection queries for a finite cylinder and an aligned box or oriented box. Translations, rotations and reflections do not change the outcome, so the boxes are transformed to canonical boxes by such transformations. This leads to simpler formulation of the test-intersection queries.

1.1 Finite Cylinder

The *finite cylinder* has center $\boldsymbol{C}$, unit-length axis direction $\boldsymbol{D}$, radius r and height h. The cylinder axis is the line segment $\boldsymbol{C}+t\boldsymbol{D}$ for $t \in [-h/2, h/2]$. The planes containing the cylinder end disks are $\boldsymbol{D}\cdot(\boldsymbol{X}-\boldsymbol{C}) = \pm h/2$. The two planes bound an infinite *slab* with points $\boldsymbol{X}$ for which $-h/2 \le \boldsymbol{D}\cdot(\boldsymbol{X}-\boldsymbol{C}) \le h/2$. The bounding planes will be referred to as *slab planes*.

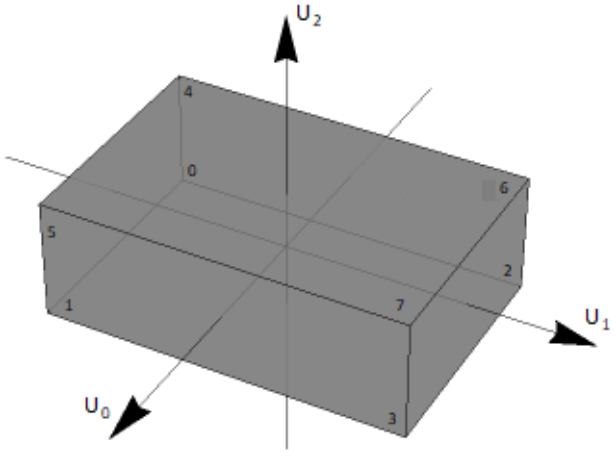
1.2 Canonical Box

The coordinate axis directions are $\boldsymbol{U}_0 = (1,0,0)$, $\boldsymbol{U}_1 = (0,1,0)$ and $\boldsymbol{U}_2 = (0,0,1)$. A *canonical box* is aligned with the coordinate axes and has center at the origin $(0,0,0)$. The extents are $\boldsymbol{E} = (e_0, e_1, e_2)$ whose components are positive. The vertices of the box are

$$\begin{aligned} &\boldsymbol{V}_0 = (-e_0,-e_1,-e_2), \quad \boldsymbol{V}_1 = (+e_0,-e_1,-e_2), \quad \boldsymbol{V}_2 = (-e_0,+e_1,-e_2), \quad \boldsymbol{V}_3 = (+e_0,+e_1,-e_2) \\ &\boldsymbol{V}_4 = (-e_0,-e_1,+e_2), \quad \boldsymbol{V}_5 = (+e_0,-e_1,+e_2), \quad \boldsymbol{V}_6 = (-e_0,+e_1,+e_2), \quad \boldsymbol{V}_7 = (+e_0,+e_1,+e_2) \end{aligned} \tag{1}$$

Figure 1 shows a canonical box with vertices labeled by their indices.

Figure 1. A canonical box with labeled vertices. The figure was drawn using Mathematica [4].



The index i for $\boldsymbol{V}_i$ is chosen so that $i = b_2b_1b_0$, where the b_j are bits 0 or 1. The generic equation for a vertex is

$$\boldsymbol{V}_i = \boldsymbol{V}_{b_2b_1b_0} = ((2b_0 - 1)e_0, (2b_1 - 1)e_1, (2b_2 - 1)e_2) \tag{2}$$

The aligned or oriented boxes can be rigidly transformed to canonical boxes. It then suffices to create a test-intersection query for a canonical box and a finite cylinder.

1.3 Aligned Box

An *aligned box* is defined by

$$\boldsymbol{X} \in [\boldsymbol{B}_{\min}, \boldsymbol{B}_{\max}] \tag{3}$$

where $\boldsymbol{B}_{\min} < \boldsymbol{B}_{\max}$. A point $\boldsymbol{X}$ in the box satisfies $\boldsymbol{B}_{\min} \le \boldsymbol{X} \le \boldsymbol{B}_{\max}$. The comparisons are performed componentwise.

The aligned box can be translated to a canonical box by subtracting its center $\boldsymbol{K} = (\boldsymbol{B}_{\max} + \boldsymbol{B}_{\min})/2$. The extents of the translated box are $\boldsymbol{E} = (\boldsymbol{B}_{\max} - \boldsymbol{B}_{\min})/2$. The cylinder center $\boldsymbol{C}$ becomes $\boldsymbol{C} - \boldsymbol{K}$.

1.4 Oriented Box

An *oriented box* is defined by a center $\boldsymbol{K}$, a right-handed orthonormal set of axis directions $\{\boldsymbol{W}_0, \boldsymbol{W}_1, \boldsymbol{W}_2\}$ and corresponding extents $\boldsymbol{E} = (e_0, e_1, e_2)$. A box point $\boldsymbol{X}$ is reprented by

$$\boldsymbol{X} = \boldsymbol{K} + \sum_{i=0}^{2} \xi_i \boldsymbol{W}_i = \boldsymbol{K} + R\boldsymbol{\xi} \tag{4}$$

for $|\xi_i| \le e_i$. The matrix $R = [\boldsymbol{W}_0\, \boldsymbol{W}_1\, \boldsymbol{W}_2]$ is a rotation matrix whose columns are the axis directions. The vector $\boldsymbol{\xi} = [\xi_0\, \xi_1\, \xi_2]^{\mathsf{T}}$ whose rows are the coefficients in the displayed equation.

The oriented box can be translated and rotated to a canonical box by subtracting its center $\boldsymbol{K}$ and rotating its axis directions $\boldsymbol{W}_i$ to the standard directions $\boldsymbol{U}_i$. The points $\boldsymbol{\xi}$ are in the canonical box. The cylinder center becomes $R^{\mathsf{T}}(\boldsymbol{C} - \boldsymbol{K})$ and the cylinder axis direction becomes $R^{\mathsf{T}}\boldsymbol{D}$.

1.5 Abstract Formulation of the Test-Intersection Query

The inputs to the query are a canonical box and a finite cylinder. The goal is to search the canonical box for a point $\widehat{\boldsymbol{X}}$ that is closest to the cylinder axis segment $\boldsymbol{C} + s\boldsymbol{D}$ for $s \in [-h/2, h/2]$. The box and cylinder intersect when the distance from $\widehat{\boldsymbol{X}}$ to the cylinder axis segment is no larger than the cylinder radius.

The candidates for $\widehat{\boldsymbol{X}}$ are points $\boldsymbol{X}$ inside the box and inside the slab. This set is a convex polyhedron defined by the linear inequalities

$$\begin{aligned} &-h/2 \le \boldsymbol{D} \cdot (\boldsymbol{X} - \boldsymbol{C}) \le h/2, && \text{slab plane constraints} \\ &-e_i \le \boldsymbol{U}_i \cdot \boldsymbol{X} \le e_i,\ 0 \le i \le 2, && \text{box constraints} \end{aligned} \tag{5}$$

Let $\boldsymbol{X}$ be a point in the box and let $\boldsymbol{Y}$ be a point in the finite cylinder. Minimizing the distance $|\boldsymbol{X}-\boldsymbol{Y}|$ is equivalent to minimizing half the squared distance

$$F(\boldsymbol{X},\boldsymbol{Y}) = \frac{1}{2}\,|\boldsymbol{X}-\boldsymbol{Y}|^2 \tag{6}$$

subject to the constraints (5). Section 2 describes an algorithm to search the constrained domain for a minimizer of F.

Section 3 is based on clipping the box by the cylinder slab planes to obtain a convex polyhedron, projecting the vertices of that polyhedron onto the plane $\boldsymbol{D}\cdot(\boldsymbol{X}-\boldsymbol{C})=0$, computing the convex hull of the projected vertices to produce a convex polygon and then computing the distance from the cylinder center to the polygon and comparing it to the cylinder radius.

Section 4 is based on a detailed analysis of the inequalities (5). The algorithm is similar to the convex hull approach except that the convex polyhedron and the convex polygon are not computed explicitly.

1.6 Early Exit via Box Culling

A separation algorithm can be used to determine when the canonical box is outside the slab that contains the cylinder. In this case the canonical box cannot intersect the cylinder.

The canonical box is projected onto the cylinder axis $\boldsymbol{L}(t)=\boldsymbol{C}+t\boldsymbol{D}$, creating a t-interval t-interval $I=[t_{\min},t_{\max}]$. The projection values are $t=\boldsymbol{D}\cdot(\boldsymbol{X}-\boldsymbol{C})$. For each $t\in I$, there is at least one box point that projects to the axis point $\boldsymbol{L}(t)$. For each $t\notin I$, there are no box points that project to $\boldsymbol{L}(t)$. The cylinder itself projects to the t-interval $[-h/2,h/2]$.

The canonical box has points $\boldsymbol{X}=(x_0,x_1,x_2)$ with $|x_i|\le e_i$. The projections onto the cylinder axis are

$$\begin{aligned} t &= \boldsymbol{D}\cdot(\boldsymbol{X}-\boldsymbol{C}) \\ &= \boldsymbol{D}\cdot\boldsymbol{X}-\boldsymbol{D}\cdot\boldsymbol{C} \\ &= \textstyle\sum_{i=0}^{2} d_i x_i - \boldsymbol{D}\cdot\boldsymbol{C} \end{aligned} \tag{7}$$

Observe that $-|d_i|e_i \le d_i x_i \le |d_i|e_i$. Define $\gamma=-\boldsymbol{D}\cdot\boldsymbol{C}$ and $\rho=|d_0|e_0+|d_1|e_1+|d_2|e_2$. The projection interval is $[t_{\min},t_{\max}]=[\gamma-\rho,\gamma+\rho]$.

The box is outside the cylinder slab when $[\gamma-\rho,\gamma+\rho]$ and $[-h/2,h/2]$ do not intersect. The conditions for nonintersection are $\gamma-\rho>h/2$ or $\gamma+\rho<-h/2$. These can be rewritten as $\gamma>\rho+h/2$ or $-\gamma>\rho+h/2$, which in turn can be rewritten as the single comparison $|\gamma|>\rho+h/2$. Listing 1 contains pseudocode for the box culling.

Listing 1. Pseudocode for the early-exit test for box culled against cylinder slab planes.

```
bool BoxIsOutsideCylinderSlab(Vector3 C, Vector3 D, Real h, Vector3 E)
{
    // Test whether the box is outside the slab contained by the planes of the cylinder end disks.
    // This is accomplished by computing the interval of projection of the box onto the cylinder axis.
    Vector3 absD{ abs(D[0]), abs(D[1]), abs(D[2]) };
    Real p = Dot(absD, E);
    Real gamma = -Dot(D, C);
    if (abs(gamma) > p + h/2)
```

```
    {
        // The box does not intersect the slab, so it does not intersect the cylinder.
        return true;
    }
    else
    {
        // The box intersects the slab. At this time it is unknown whether the box intersects the cylinder.
        return false;
    }
}
```

---

2 Algorithm Based on Convex Quadratic Programming

In *Robust and Error-Free Geometric Computing* [1], I discuss the convex quadratic programming problem (CQP) and a general method of solving these by transforming them to the framework of a linear complementarity problem (LCP).

The quadratic programming problem is the following. Given constants $A \in \mathbb{R}^{n\times n}$, $\boldsymbol{b} \in \mathbb{R}^n$, $c \in \mathbb{R}$, $D \in \mathbb{R}^{m\times n}$, $\boldsymbol{e} \in \mathbb{R}^m$ and variable $\boldsymbol{x} \in \mathbb{R}^n$, minimize $f(\boldsymbol{x}) = \frac{1}{2}\boldsymbol{x}^{\mathsf{T}}A\boldsymbol{x} + \boldsymbol{b}^{\mathsf{T}}\boldsymbol{x} + c$ subject to the linear inequality constraints $\boldsymbol{x} \geq \boldsymbol{0}$ and $D\boldsymbol{x} \geq \boldsymbol{e}$. The number of linear inequality constraints is $n + m$. The problem is referred to as a convex quadratic programming problem when the matrix A is positive definite or positive semidefinite.

The linear complementarity problem is the following. Given constants $\boldsymbol{q} \in \mathbb{R}^k$ and $M \in \mathbb{R}^{k\times k}$, find $\boldsymbol{z} \in \mathbb{R}^k$ such that $\boldsymbol{z} \geq \boldsymbol{0}$, $\boldsymbol{q} + M\boldsymbol{z} \geq \boldsymbol{0}$ and $\boldsymbol{z}^{\mathsf{T}}(\boldsymbol{q} + M\boldsymbol{z}) = \boldsymbol{0}$. Define $\boldsymbol{w} = \boldsymbol{q} + M\boldsymbol{z}$. Choose $\boldsymbol{z} \geq \boldsymbol{0}$ such that $\boldsymbol{w} \geq \boldsymbol{0}$ and $\boldsymbol{z}^{\mathsf{T}}\boldsymbol{w} = 0$.

The CQP is transformed to an LCP by defining

$$\boldsymbol{q} = \begin{bmatrix} \boldsymbol{b} \\ -\boldsymbol{e} \end{bmatrix}, \quad M = \begin{bmatrix} A & -D^{\mathsf{T}} \\ D & 0 \end{bmatrix}, \quad \boldsymbol{z} = \begin{bmatrix} \boldsymbol{x} \\ \boldsymbol{y} \end{bmatrix}, \quad \boldsymbol{w} = \begin{bmatrix} \boldsymbol{u} \\ \boldsymbol{v} \end{bmatrix} \tag{8}$$

where $k = n + m$. The matrix M is not symmetric, but it is positive semidefinite because $\boldsymbol{z}^{\mathsf{T}}M\boldsymbol{z} \geq 0$ for all $\boldsymbol{z}$. The inequality is guaranteed because A is positive semidefinite.

The Geometric Tools source code has an LCP solver (file LCPSolver.h) that supports floating-point arithmetic, exact rational arithmetic and quadratic-field arithmetic.

For the CQP representing the test-intersection query for a canonical box and a finite cylinder, the vector and matrix quantities are derived next.

The variable naming to match the specific problem to the general CQP formulation requires special attention. Let the canonical box have point $\boldsymbol{\xi} \in [-\boldsymbol{\ell}, \boldsymbol{\ell}]$, where $\boldsymbol{\ell} = (\ell_0, \ell_1, \ell_2)$ are the positive extents. To satisfy the nonnegativity constraints $\boldsymbol{x} \geq \boldsymbol{0}$ in the CQP statement, we need to translate the box, $\boldsymbol{x} = \boldsymbol{\xi} + \boldsymbol{\ell}$. The constraints are now $\boldsymbol{0} \leq \boldsymbol{x} \leq 2\boldsymbol{\ell}$. Let the cylinder center be $\boldsymbol{c} = (c_0, c_1, c_2)$ and the cylinder axis direction be $\boldsymbol{w} = (w_0, w_1, w_2)$. The cylinder must also be translated, so its new center is $\boldsymbol{k} = \boldsymbol{c} + \boldsymbol{\ell}$.

The linear inequality constraints $D\boldsymbol{x} \geq \boldsymbol{e}$ include the box constraints $\boldsymbol{x} \leq 2\boldsymbol{\ell}$ and the slab constraints $-h/2 \leq \boldsymbol{w} \cdot (\boldsymbol{x} - \boldsymbol{k}) \leq h/2$. These are 5 constraints in 3 unknowns, so $n = 3$ and $m = 5$ in the CQP

formulation. The 5×3 matrix D and the 5×1 vector $\boldsymbol{e}$ are

$$D = \begin{bmatrix} -1 & 0 & 0 \\ 0 & -1 & 0 \\ 0 & 0 & -1 \\ w_0 & w_1 & w_2 \\ -w_0 & -w_1 & -w_2 \end{bmatrix}, \quad \boldsymbol{e} = \begin{bmatrix} -2\ell_0 \\ -2\ell_1 \\ -2\ell_2 \\ w_0k_0 + w_1k_1 + w_2k_2 - h/2 \\ -w_0k_0 - w_1k_1 - w_2k_2 - h/2 \end{bmatrix} \tag{9}$$

For a point $\boldsymbol{x}$ in the box, half the squared distance between $\boldsymbol{x}$ and the cylinder is half the squared length of the projection of $\boldsymbol{x} - \boldsymbol{k}$ onto the plane containing $\boldsymbol{k}$ and having normal $\boldsymbol{w}$. The projection is

$$(\boldsymbol{x} - \boldsymbol{k}) - (\boldsymbol{w} \cdot (\boldsymbol{x} - \boldsymbol{k}))\boldsymbol{w} = \left(I - \boldsymbol{w}\boldsymbol{w}^{\mathsf{T}}\right)(\boldsymbol{x} - \boldsymbol{k}) \tag{10}$$

Half the squared length is

$$f(\boldsymbol{x}) = \frac{1}{2}(\boldsymbol{x} - \boldsymbol{k})^{\mathsf{T}}\left(I - \boldsymbol{w}\boldsymbol{w}^{\mathsf{T}}\right)(\boldsymbol{x} - \boldsymbol{k}) = \frac{1}{2}\boldsymbol{x}^{\mathsf{T}}\left(I - \boldsymbol{w}\boldsymbol{w}^{\mathsf{T}}\right)\boldsymbol{x} - \boldsymbol{k}^{\mathsf{T}}\left(I - \boldsymbol{w}\boldsymbol{w}^{\mathsf{T}}\right)\boldsymbol{x} + \frac{1}{2}\boldsymbol{k}^{\mathsf{T}}\left(I - \boldsymbol{w}\boldsymbol{w}^{\mathsf{T}}\right)\boldsymbol{k} \tag{11}$$

The 3×3 matrix A and the 3×1 vector $\boldsymbol{b}$ are

$$A = I - \boldsymbol{w}\boldsymbol{w}^{\mathsf{T}}, \quad \boldsymbol{b} = \left(I - \boldsymbol{w}\boldsymbol{w}^{\mathsf{T}}\right)\boldsymbol{k} \tag{12}$$

The constant of the quadratic equation is irrelevant for computing a minimizer $\boldsymbol{x}$, although technically it is required for computing the minimum of f at the minimizer $\boldsymbol{x}$.

The quantities $\boldsymbol{q}$ and M of equation (8) are computed from A, $\boldsymbol{b}$, D and $\boldsymbol{e}$. The LCP solver inputs are the 8×8 matrix M and the 8×1 vector $\boldsymbol{q}$. The outputs are the 8×1 vectors $\boldsymbol{w}$ and $\boldsymbol{z}$. The minimizer $\boldsymbol{x}$ of f consists of the first three components of $\boldsymbol{z}$, namely, $\boldsymbol{x} = (z_0, z_1, z_2)$.

The squared distance from the minimizer to the cylinder is $d^2 = (\boldsymbol{x} - \boldsymbol{k})^{\mathsf{T}}\left(I - \boldsymbol{w}\boldsymbol{w}^{\mathsf{T}}\right)(\boldsymbol{x} - \boldsymbol{k})$. The box and the cylinder intersect when $d^2 \le r^2$ where r is the cylinder radius.

The LCP approach allows for rational arithmetic in the query, but note that if $\boldsymbol{D}$ is not exactly unit length (computed with floating-point and passed to the query), the query can be theoretically incorrect because the cylinder representation itself suffers from rounding errors. This can be remedied using rational quadratic fields whose numbers are of the form $x + y|\boldsymbol{D}| = x + y\sqrt{\boldsymbol{D} \cdot \boldsymbol{D}}$ where x, y and $\boldsymbol{D}$ are rational valued. The Geometric Tools LCP solver supports rational quadratic fields.

The LCP approach suffers from numerical robustness problems, however, when floating-point arithmetic is used. This makes the approach undesirable in practice if the user chooses not to pay the cost of rational arithmetic.

3 Algorithm Based on Convex Hulls

The construction in this section assumes that the early-exit culling step has been applied whereby the box is not outside the cylinder slab.

The canonical box is clipped by the cylinder slab planes to form a convex polyhedron. The projection of the convex polyhedron onto the plane $\boldsymbol{D} \cdot (\boldsymbol{X} - \boldsymbol{C})$ is a convex polygon. The point $\boldsymbol{C}$ is transformed to the

origin $\mathbf{0}$. If the distance from the origin to the solid convex polygon is no larger than r, the box and cylinder intersect.

The straightforward implementation is to compute the set S of box vertices that are inside or on the cylinder slab. If $\boldsymbol{V}$ is a box vertex and $t = \boldsymbol{D} \cdot (\boldsymbol{V} - \boldsymbol{C}) \in [-h/2, h/2]$, then $\boldsymbol{V} \in S$. The intersection of the slab planes with the box edges are computed and inserted into S. The set S is projected to the aforementioned plane. The convex polygon in the plane is the convex hull of the projected S points.

Note that in some configurations, S contains points that are not convex hull vertices. It is necessary to use a generic 2D convex hull algorithm to compute the convex polygon from S. The number of elements of S is small, so an algorithm such as Gift wrapping, Graham scan or Andrew's algorithm can be used [2].

Once the convex polygon is computed as an ordered list of 2-tuples, if $(0, 0)$ is inside or on the polygon, the cylinder and box intersect. If $(0, 0)$ is outside the polygon, the squared distance δ^2 from $(0, 0)$ to the polygon is computed. The cylinder and box intersect when $\delta^2 \le r^2$.

4 Algorithm Based on Linear Inequalities

The construction in this section assumes that the early-exit culling step has been applied whereby the box is not outside the cylinder slab.

Reflection transformations do not affect the test-intersection query. If the query is true in a coordinate system, the query is still true if one or more variables of the coordinate system are reflected. This allows us to derive the construction for cylinder axis direction vectors $\boldsymbol{D} = (d_0, d_1, d_2)$ for which $d_i \ge 0$; the directions are in the first octant. The directions can be partitioned into those with 0, 1 or 2 zero-valued compoments, which effectively reduces the dimensionality of the problem per zero-valued component.

A reflection in the i-th coordinate is performed when $d_i < 0$, leading to a new component with $d_i' = -d_i > 0$. The same reflection must be applied to the cylinder center; that is, $c_i' = -c_i$. The reflection for the canonical box does not require any sign changes because the box is symmetric in each coordinate plane.

Without reflections, the partitioning of directions based on signs of components involves 26 cases. Each component has 3 signs to analyze—positive, negative or zero—which implies $3^3 = 27$ cases to analyze. However, the case $\boldsymbol{D} = (0, 0, 0)$ is irrelevant because the direction cannot be the zero vector. With the reflections, the partitioning of directions based on signs of components involves 7 cases. Each component has 2 signs to analyze—positive or zero—which implies $2^3 = 8$ cases to analyze. The case $\boldsymbol{D} = (0, 0, 0)$ is irrelevant.

The problem is to determine whether the finite cylinder intersects the convex polyhedron that is the canonical box clipped by the cylinder slab planes. Points $\boldsymbol{X}$ in the convex polyhedron are defined by the 8 linear inequalities

$$|\boldsymbol{U}_0 \cdot \boldsymbol{X}| \le e_0, \;\; |\boldsymbol{U}_1 \cdot \boldsymbol{X}| \le e_1, \;\; |\boldsymbol{U}_2 \cdot \boldsymbol{X}| \le e_2, \;\; |\boldsymbol{D} \cdot (\boldsymbol{X} - \boldsymbol{C})| \le h/2 \tag{13}$$

4.1 Directions with Two Zero-Valued Components

Let (i_0, i_1, i_2) be a permutation of $(0, 1, 2)$ for which $d_{i_0} > 0$, $d_{i_1} = 0$ and $d_{i_2} = 0$. Because $\boldsymbol{D}$ is unit length and has nonnegative components, it must be that $d_{i_0} = 1$. Points can be represented as $\boldsymbol{X} =$

$\boldsymbol{C} + t\boldsymbol{D} + (x_{i_1} - c_{i_2})\boldsymbol{U}_{i_1} + (x_{i2} - c_{i_2})\boldsymbol{U}_{i_2}$. Substituting $\boldsymbol{X}$ into the constraints (13),

$$|c_{i_0} + td_{i_0}| = |c_{i_0} + t| \le e_{i_0}, \;\; |x_{i_1}| \le e_{i_1}, \;\; |x_{i_2}| \le e_{i_2}, \;\; t \le |h/2| \tag{14}$$

The two inequalities (14) that have a t-variable can be manipulated to inequalities that isolate the t-variable,

$$-e_{i_0} - c_{i_0} \le t \le e_{i_0} - c_{i_0}, \;\; -h/2 \le t \le h/2 \tag{15}$$

Combining these into a single pair of inequalities,

$$\ell = \max\{-e_{i_0} - c_{i_0}, -h/2\} \le t \le \min\{e_{i_0} - c_{i_0}, h/2\} = \mu \tag{16}$$

where the first equality defines ℓ and the last equality defines μ. The expression $\ell \le \mu$ is true when $(\ell, \mu) \in \{(-e_{i_0} - c_{i_0}, e_{i_0} + c_{i_0}), (-h/2, h/2)\}$. If $(\ell, \mu) = (-e_{i_0} - c_{i_0}, h/2)$, the expression is true when $c_0 \ge -e_0 - h/2$. If $(\ell, \mu) = (-h/2, e_{i_0} - c_{i_0})$, the expression is true when $c_0 \le e_0 + h/2$. Assuming that the box culling of Listing 1 is attempted first in the query, the expression $\ell \le \mu$ is always true for the 2-zero-component case.

4.1.1 Early Exit via Axis-Box Intersection Testing

The projection of the cylinder axis $\boldsymbol{C} + t\boldsymbol{D}$ onto the plane containing $\boldsymbol{C}$ and having normal vector $\boldsymbol{D}$ is $(x_{i_1}, x_{i_2}) = (c_{i_1}, c_{i_2})$. The cylinder axis intersects the convex polyhedron when the following inequalites are true,

$$|c_{i_1}| \le e_{i_1}, \;\; |c_{i_2}| \le e_{i_2} \tag{17}$$

and, if true, the cylinder and box intersect.

4.1.2 Test Intersection via Projection and Distance.

If $|c_{i_1}| > e_{i_1}$ or $|c_{i_2}| > e_{i_2}$, the cylinder axis does not intersecct the box. The projection of the cylinder axis is outside the projection of the box. The cylinder and box intersect when the distance from $(0, 0)$ to the box is no larger than the cylinder radius. The squared distance from a 2D point to a solid 2D canonical rectangle is simple to compute.

Listing 2 contains pseudocode for the 2-zero-component case.

Listing 2. Pseudocode to determine whether the box and cylinder intersect when the direction has 2 zero-valued components d_{i_1} and d_{i_2} with $d_{i_0} = 1$. The input array i contains (i_0, i_1, i_2).

```
bool Intersects2(array<int, 3> i, Vector3 C, Real r, Vector3 E)
{
    // The 2-tuple (c_{i1}, c_{i2}) is the projected cylinder axis. The 2-tuple (e_{i1}, e_{i2}) is the extent of the projected canonical box
    // which is an axis-aligned rectangle.
    Real absC1 = abs(C[i[1]]), absC2 = abs(C[i[2]]), E1 = E[i[1]], E2 = E[i[2]];

    // Test whether the cylinder axis and canonical box intersect.
    if (absC1 <= E1 && absC2 <= E2)
    {
        return true;
    }
```

```
    // Compute the squared distance from the projected cylinder axis to the projected canonical box.
    Real sqrDistance = 0;
    Real delta = absC1 - E1;
    if (delta > 0)
    {
        sqrDistance += delta * delta;
    }
    delta = absC2 - E2;
    if (delta > 0)
    {
        sqrDistance += delta * delta;
    }
    return sqrDistance <= r * r;
}
```

4.2 Directions with One Zero-Valued Component

Let (i_0, i_1, i_2) be a permutation of $(0, 1, 2)$ for which $d_{i_0} > 0$, $d_{i_1} > 0$ and $d_{i_2} = 0$. Because $\boldsymbol{D}$ is unit length, it must be that $d_{i_0}^2 + d_{i_1}^2 = 1$. Points can be represented as $\boldsymbol{X} = \boldsymbol{C} + t\boldsymbol{D} + s\boldsymbol{D}^\perp + (x_{i_2} - c_{i_2})\boldsymbol{U}_{i_2}$ where $\boldsymbol{D}^\perp$ is a vector perpendicular to $\boldsymbol{D}$ based on the index permutation. Specifically, it is $(d_1, -d_0, 0) = \boldsymbol{D} \times \boldsymbol{U}_2$ for $i_2 = 2$, $(d_2, 0, -d_0) = -\boldsymbol{D} \times \boldsymbol{U}_1$ for $i_2 = 1$ or $(0, d_2, -d_1) = \boldsymbol{D} \times \boldsymbol{U}_0$ for $i_2 = 0$. Substituting $\boldsymbol{X}$ into the constraints (13),

$$|c_{i_0} + td_{i_0} + sd_{i_1}| \le e_{i_0}, \;\; |c_{i_1} + td_{i_1} - sd_{i_0}| \le e_{i_1}, \;\; |x_{i_2}| \le e_{i_2}, \;\; |t| \le h/2 \tag{18}$$

4.2.1 Early Exit via Axis-Box Intersection Testing

The three inequalities (18) that have a t-variable can be manipulated to inequalities that isolate the t-variable,

$$\begin{aligned} \alpha_0(s) &= (-e_{i_0} - c_{i_0} - sd_{i_1})/d_{i_0} \le t \le (e_{i_0} - c_{i_0} - sd_{i_1})/d_{i_0} = \beta_0(s) \\ \alpha_1(s) &= (-e_{i_1} - c_{i_1} + sd_{i_0})/d_{i_1} \le t \le (e_{i_1} - c_{i_1} + sd_{i_0})/d_{i_1} = \beta_1(s) \end{aligned} \tag{19}$$

where the first and last equalities define $\alpha_i(s)$ and $\beta_i(s)$. Combining these into a single pair of inequalities,

$$\ell(s) = \max\{\alpha_0(s), \alpha_1(s), -h/2\} \le t \le \min\{\beta_0(s), \beta_1(s), h/2\} = \mu(s) \tag{20}$$

where the first and equalities defines $\ell(s)$ and $\mu(s)$.

The projection plane contains $\boldsymbol{C}$ and has normal $\boldsymbol{D}$. The inequality $\ell(s) \le \mu(s)$ and constraints $|x_{i_2}| \le e_{i_2}$ define the projection of the convex polyhedron. The projection is a rectangle which is a convex polygon.

The projection of the cylinder axis $\boldsymbol{C} + t\boldsymbol{D}$ onto the plane is $(s, x_{i_2}) = (0, c_{i_2})$. The cylinder axis intersects the convex polyhedron when $\ell(0) \le \mu(0)$ and $|c_{i_2}| \le e_{i_2}$, which reduce to

$$\max\left\{\frac{-e_{i_0} - c_{i_0}}{d_{i_0}}, \frac{-e_{i_1} - c_{i_1}}{d_{i_1}}, -\frac{h}{2}\right\} \le \min\left\{\frac{e_{i_0} - c_{i_0}}{d_{i_0}}, \frac{e_{i_1} - c_{i_1}}{d_{i_1}}, \frac{h}{2}\right\}, \;\; |c_{i_2}| \le e_{i_2} \tag{21}$$

Listing 3 contains pseudocode for the early-exit test. If $d_i > 0$ is a subnormal floating-point number and the division of one of $(\pm e_i - c_i)/d_i$ overflows, the computation of extreme values and the comparison of $\ell(0)$ and $\mu(0)$ are still correct.

Listing 3. Pseudocode for the early-exit test for the intersection of the cylinder axis and the box when $\boldsymbol{D}$ has one zero-valued component d_{i_2} and the components d_{i_0} and d_{i_1} are positive. The input array i contains (i_0, i_1, i_2). It is assumed that the early-exit culling step has been applied previously, so when the function is called, the box is not outside the cylinder slab.

```
bool CylinderAxisIntersectsBox2D(array<int, 3> i, Vector3 C, Vector3 D, Real hDiv2, Vector3 E)
{
    if (abs(C[i[2]]) <= E[i[2]])
    {
        array<Real, 2> negEmCDivD = { (-E[i[0]] - C[i[0]]) / D[i[0]], (-E[i[1]] - C[i[1]]) / D[i[1]] };
        array<Real, 2> posEmCDivD = { (+E[i[0]] - C[i[0]]) / D[i[0]], (+E[i[1]] - C[i[1]]) / D[i[1]] };
        Real lower = max(max(negEmCDivD[0], negEmCDivD[1]), -hDiv2);
        Real upper = min(min(posEmCDivD[0], posEmCDivD[1]), +hDiv2);
        return lower <= upper;
    }
    return false;
}
```

In the next section it is assumed that the early-exit test failed and that the cylinder axis does not intersect the box.

4.2.2 Test Intersection via Projection and Distance

The inequalities (18) that contain s can be manipulated to inequalities that isolate the s-variable,

$$\begin{aligned} \alpha_0(t) &= (-e_{i_0} - c_{i_0} - td_0)/d_1 \le s \le (e_{i_0} - c_{i_0} - td_0)/d_1 = \beta_0(t) \\ \alpha_1(t) &= (-e_{i_1} + c_{i_1} + td_1)/d_0 \le s \le (e_{i_1} + c_{i_1} + td_1)/d_0 = \beta_1(t) \end{aligned} \tag{22}$$

where the first equalities define $\alpha_i(t)$ and the last equalities define $\beta_i(t)$. These four constraints can be algebraically manipulated to obtain

$$\alpha_0(t) \le s \le \beta_0(t), \quad \alpha_1(t) \le s \le \beta_1(t) \tag{23}$$

for linear functions $\alpha_i(t)$ and $\beta_i(t)$. The slopes of $\alpha_0(t)$ and $\beta_0(t)$ are negative and the slopes of $\alpha_1(t)$ and $\beta_1(t)$ are positive. The combined constraints are

$$\ell(t) = \max\{\alpha_0(t), \alpha_1(t)\} \le s \le \min\{\beta_0(t), \beta_1(t)\} = \mu(t) \tag{24}$$

where the first equality defines $\ell(t)$ and the last equality defines $\mu(t)$. Figure 2 shows typical graphs of two linear function and their maximum function and minimum function.

Figure 2. The function $f(t)$ is a line with negative slope and the function $g(t)$ is a function with positive slope. The minimum function $\min\{f(t), g(t)\}$ is drawn in blue. The maximum function $\max\{f(t), g(t)\}$ is drawn in gold. The minimum of $\max\{f(t), g(t)\}$ and the maximum of $\min\{f(t), g(t)\}$ occur at the t-value where $f(t) = g(t)$. The figure was drawn with Mathematica [4].

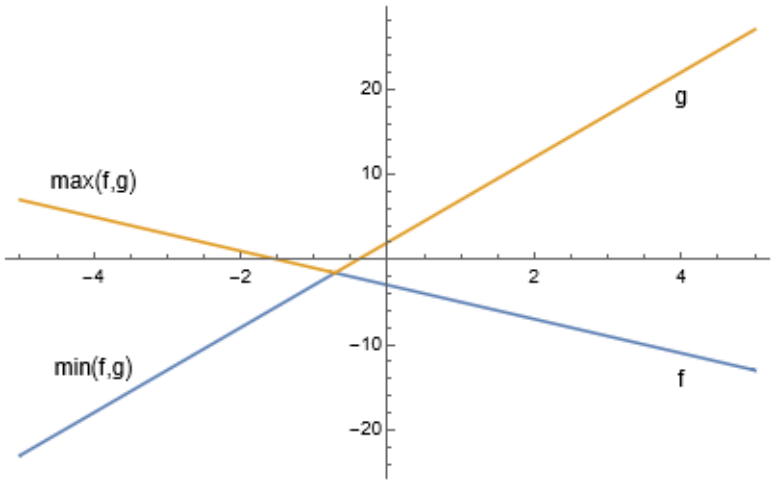


Figure 2 is drawn as if t has no constraints. However, we have the inequality constraints $t \in [-h/2, h/2]$. The minimum and maximum s-values over all t are

$$s_{\min} = \min_{|t|\le h/2} \ell(t) \le s \le \max_{|t|\le h/2} \mu(t) = s_{\max} \tag{25}$$

Let $\hat{t}$ be the value for which $\alpha_0(\hat{t}) = \alpha_1(\hat{t})$, namely, $\hat{t} = d_{i_0}(-e_{i_0} - c_{i_0}) + d_{i_1}(e_{i_1} - c_{i_1})$; then

$$s_{\min} = \left\{ \begin{array}{ll} \alpha_0(\hat{t}), & \hat{t} \in [-h/2, +h/2] \\ \alpha_1(-h/2), & \hat{t} < -h/2 \\ \alpha_0(+h/2), & \hat{t} > +h/2 \end{array} \right\} = \left\{ \begin{array}{ll} -d_{i_0}(e_{i_1} - c_{i_1}) + d_{i_1}(-e_{i_0} - c_{i_0}), & \hat{t} \in [-h/2, +h/2] \\ (-e_{i_1} + c_{i_1} - d_{i_1}h/2)/d_{i_0}, & \hat{t} < -h/2 \\ (-e_{i_0} - c_{i_0} - d_{i_0}h/2)/d_{i_1}, & \hat{t} > +h/2 \end{array} \right\} \tag{26}$$

Let $\bar{t}$ be the value for which $\beta_0(\bar{t}) = \beta_1(\bar{t})$, namely, $\bar{t} = d_{i_0}(e_{i_0} - c_{i_0}) + d_{i_1}(-e_{i_1} - c_{i_1})$; then

$$s_{\max} = \left\{ \begin{array}{ll} \beta_0(\bar{t}), & \bar{t} \in [-h/2, +h/2] \\ \beta_0(-h/2), & \bar{t} < -h/2 \\ \beta_1(+h/2), & \bar{t} > +h/2 \end{array} \right\} = \left\{ \begin{array}{ll} -d_{i_0}(-e_{i_1} - c_{i_1}) + d_{i_1}(e_{i_0} - c_{i_0}), & \bar{t} \in [-h/2, +h/2] \\ (e_{i_0} - c_{i_0} + d_{i_0}h/2)/d_{i_1}, & \bar{t} < -h/2 \\ (e_{i_1} + c_{i_1} + d_{i_1}h/2)/d_{i_0}, & \bar{t} > +h/2 \end{array} \right\} \tag{27}$$

The projection of the convex polyhedron are those parameterized points $\boldsymbol{X}$ for which $(s, x_{i_2}) \in [s_{\min}, s_{\max}] \times [-e_{i_2}, e_{i_2}]$, which is a rectangle. The cylinder axis in the projection plane is $(s, x_{i_2}) = (0, c_{i_2})$. Assuming the early-exit test was applied previously, the projection point is outside the projection rectangle. The canonical box and finite cylinder intersect when the distance from $(0, c_{i_2})$ to the rectangle is no larger than the cylinder radius.

Listing 4 contains pseudocode that works for directions that have 1 zero-valued component.

Listing 4. Pseudocode for the test-intersection query of the cylinder axis and the box when $\boldsymbol{D}$ has one zero-valued component d_{i_2} and the components d_{i_0} and d_{i_1} are positive. The input array i contains (i_0, i_1, i_2). It is assumed that the early-exit culling step has been applied previously, so when the function is called, the box is not outside the cylinder slab.

```
bool Intersects1(array<int, 3> i, Vector3 C, Vector3 D, Real r, Real hDiv2, Vector3 E)
{
    Real c0 = C[i[0]], c1 = C[i[1]], c2 = C[i[2]];
    Real d0 = D[i[0]], d1 = D[i[1]];
    Real e0 = E[i[0]], e1 = E[i[1]], e2 = E[i[2]];
    Real e0pc0 = e0 + c0, e0mc0 = e0 - c0, e1pc1 = e1 + c1, e1mc1 = e1 - c1;

    // Test whether the cylinder axis and canonical box intersect.
    Real absC2 = abs(c2);
    if (absC2 <= e2)
    {
        array<Real, 2> negEmCDivD{ -e0pc0 / d0, -e1pc1 / d1 };
        array<Real, 2> posEmCDivD{ e0mc0 / d0, e1mc1 / d1 };
        Real lower = max(max(negEmCDivD[0], negEmCDivD[1]), -hDiv2);
        Real upper = min(min(posEmCDivD[0], posEmCDivD[1]), hDiv2);
        if (lower <= upper)
        {
            return true;
        }
    }

    // Compute the squared distance from the projected cylinder axis (a point) to the projected convex polyhedron (a rectangle).
    Real sMin = 0, tHat = d1 * e1mc1 - d0 * e0pc0;
    if (-hDiv2 <= tHat)
    {
        if (tHat <= hDiv2)
        {
            sMin = -(d0 * e1mc1 + d1 * e0pc0);
        }
        else  // tHat > +h/2
        {
            sMin = -(e1pc1 + d0 * hDiv2) / d1;
        }
    }
    else  // tHat < -h/2
    {
        sMin = -(e1mc1 + d1 * hDiv2) / d0;
    }

    Real sMax = 0, tBar = d0 * e0mc0 - d1 * e1pc1;
    if (-hDiv2 <= tBar)
    {
        if (tBar <= hDiv2)
        {
            sMax = d0 * e1pc1 + d1 * e0mc0;
        }
        else  // tBar > +h/2
        {
            sMax = (e1pc1 + d1 * hDiv2) / d0;
        }
    }
    else  // tBar < -h/2
    {
        sMax = (e0mc0 + d0 * hDiv2) / d1;
    }

    Real sqrDistance = 0;
    if (0 < sMin)
    {
        sqrDistance += sMin * sMin;
    }
    else if (sMax < 0)
    {
        sqrDistance += sMax * sMax;
    }
```

```
    Real delta = absC2 - e2;
    if (delta > 0)
    {
        sqrDistance += delta * delta;
    }

    return sqrDistance <= r * r;
}
```

4.3 Directions with No Zero-Valued Components

The direction is $\boldsymbol{D} = (d_0, d_1, d_2)$ with $d_i > 0$ for all i. The polyhedron faces are contained by up to 8 planes of the form $\boldsymbol{N} \cdot \boldsymbol{X} = c$. These are $\pm \boldsymbol{U}_0, \cdot \boldsymbol{X} = e_0$, $\pm \boldsymbol{U}_1 \cdot \boldsymbol{X} = e_1$, $\pm \boldsymbol{U}_2 \cdot \boldsymbol{X} = e_2$ and $\pm \boldsymbol{D} \cdot (\boldsymbol{X} - \boldsymbol{C}) = h/2$.

4.3.1 Early Exit via Axis-Box Intersection Testing

Parameterize points $\boldsymbol{X}$ in a coordinate system with origin $\boldsymbol{C}$ and an orthogonal set of basis vectors that includes $\boldsymbol{D}$. For the purpose of axis-box intersection testing, the basis vectors do not have to be unit length. The parameterization is

$$\boldsymbol{X} = \begin{bmatrix} c_0 \\ c_1 \\ c_2 \end{bmatrix} + t \begin{bmatrix} d_0 \\ d_1 \\ d_2 \end{bmatrix} + s_0 \begin{bmatrix} d_1 \\ -d_0 \\ 0 \end{bmatrix} + s_1 \begin{bmatrix} d_0 d_2 \\ d_1 d_2 \\ -(d_0^2 + d_1^2) \end{bmatrix} \tag{28}$$

Substituting this into the 8 plane equations,

$$\begin{aligned} -e_0 &\le c_0 + t d_0 + s_0 d_1 + s_1 d_0 d_2 \le e_0 \\ -e_1 &\le c_1 + t d_1 - s_0 d_0 + s_1 d_1 d_2 \le e_1 \\ -e_2 &\le c_2 + t d_2 - s_1 (d_0^2 + d_1^2) \le e_2 \\ -h/2 &\le t \le h/2 \end{aligned} \tag{29}$$

The inequalities can be manipulated to isolate the t-variable,

$$\begin{aligned} \alpha_0(s_0, s_1) &= (-e_0 - c_0 - s_0 d_1 - s_1 d_0 d_2)/d_0 \le t \le (e_0 - c_0 - s_0 d_1 - s_1 d_0 d_2)/d_0 = \beta_0(s_0, s_1) \\ \alpha_1(s_0, s_1) &= (-e_1 - c_1 + s_0 d_0 - s_1 d_1 d_2)/d_1 \le t \le (e_1 - c_1 + s_0 d_0 - s_1 d_1 d_2)/d_1 = \beta_1(s_0, s_1) \\ \alpha_2(s_0, s_1) &= (-e_2 - c_2 + s_1 (d_0^2 + d_1^2)/d_2 \le t \le (e_2 - c_2 + s_1 (d_0^2 + d_1^2))/d_2 = \beta_2(s_0, s_1) \\ \alpha_3(s_0, s_1) &= -h/2 \le t \le h/2 = \beta_3(s_0, s_1) \end{aligned} \tag{30}$$

where the first and last equalities define $\alpha_i(t)$ and $\beta_i(t)$. Combining these we obtain $\ell(s_0, s_1) \le t \le \mu(s_0, s_1)$ where

$$\begin{aligned} \ell(s_0, s_1) &= \max\{\alpha_0(s_0, s_1), \alpha_1(s_0, s_1), \alpha_2(s_0, s_1), \alpha_3(s_0, s_1)\} \\ \mu(s_0, s_1) &= \min\{\beta_0(s_0, s_1), \beta_1(s_0, s_1), \beta_2(s_0, s_1), \beta_3(s_0, s_1)\} \end{aligned} \tag{31}$$

The direction vectors associated with s_0 and s_1 have length $L = \sqrt{d_0^2 + d_1^2}$. The inequalities $\ell(s_0/L, s_1/L) \le \mu(s_0/L, s_1/L)$ define the convex polygon that is the projection of the convex polyhedron. Eliminating the maximum and minimum operations, there are 16 inequalities $\alpha_i(s_0/L, s_1/L) \le \beta_j(s_0/L, s_1/L)$ for $i \in \{0,1,2,3\}$ and $j \in \{0,1,2,3\}$. These can be solved using the Fourier–Motzkin algorithm [3], but this would be inefficient because redundant inequalities must be identified and discarded. The remaining inequalities are solved pairwise to generate the vertices of the convex polygon.

The projection of the cylinder axis $\boldsymbol{C} + t\boldsymbol{D}$ onto the plane containing $\boldsymbol{C}$ and having normal vector $\boldsymbol{D}$ is $(s_0, s_1) = (0, 0)$. The cylinder axis intersects the convex polyhedron when $\ell(0,0) \le \mu(0,0)$, which reduces to

$$\ell(0,0) = \max\left\{\frac{-e_0 - c_0}{d_0}, \frac{-e_1 - c_1}{d_1}, \frac{-e_2 - c_2}{d_2}, -\frac{h}{2}\right\} \le \min\left\{\frac{e_0 - c_0}{d_0}, \frac{e_1 - c_1}{d_1}, \frac{e_2 - c_2}{d_2}, \frac{h}{2}\right\} = \mu(0,0) \tag{32}$$

When this condition is true, the finite cylinder and canonical box intersect. In the next section, the assumption is that this early-exit test has been performed and the cylinder axis and box do not intersect. This means $(0, 0)$ is outside the convex polygon.

Observe that the comparisons of (32) are the same if instead the vector associated with p_0 is $(d_2, 0, -d_0)$ or $(0, d_2, -d_1)$. The implementation of the extreme values does not require a robust computation of vectors perpendicular to $\boldsymbol{D}$.

Listing 5 contains pseudocode for the early-exit test. If $d_i > 0$ is a subnormal floating-point number and the division of one of $(\pm e_i - c_i)/d_i$ overflows, the computation of extreme values and the comparison of $\ell(0,0)$ to $\mu(0,0)$ are still correct.

Listing 5. Pseudocode for the early-exit test for the intersection of the cylinder axis and the box when $\boldsymbol{D}$ has no zero-valued components. It is assumed that the early-exit culling step has been applied previously, so when the function is called, the box is not outside the cylinder slab.

```
bool CylinderAxisIntersectsBox3D(Vector3 C, Vector3 D, Real hDiv2, Vector3 E)
{
    array<Real, 3> negEmCDivD = {(-E[0] - C[0]) / D[0], (-E[1] - C[1]) / D[1], (-E[2] - C[2]) / D[2] };
    array<Real, 3> posEmCDivD = {(+E[0] - C[0]) / D[0], (+E[1] - C[1]) / D[1], (+E[2] - C[2]) / D[2] };
    Real lower = max(max(negEmCDivD[0], negEmCDivD[1]), max(negEmCDivD[2], -hDiv2));
    Real upper = min(min(posEmCDivD[0], posEmCDivD[1]), min(posEmCDivD[2], +hDiv2));
    return lower <= upper;
}
```

4.3.2 Locating Extreme Edges

The idea is to compute the extreme polyline of the convex polyhedron. The convex polygon in the plane containing $\boldsymbol{C}$ and having normal $\boldsymbol{D}$ is the projection of the extreme polyline. Each segment of the polyline is shared by two faces of the convex polyhedron. One face has outer-pointing normal $\boldsymbol{N}_0$ and the other face has outer-pointing normal $\boldsymbol{N}_1$ where $\boldsymbol{D} \cdot \boldsymbol{N}_0 > 0 > \boldsymbol{D} \cdot \boldsymbol{N}_1$.

The dot products of the normals with $\boldsymbol{D}$ are $\{\pm d_0, \pm d_1, \pm d_2, \pm 1\}$. We need only compare pairs of planes, one having a positive dot product and one having a negative dot product. The line of intersection of the pair of planes is then trimmed by the inequalities that define the convex polyhedron.

The 12 pairs of normals that can lead to extreme edges are $(\boldsymbol{U}_0, -\boldsymbol{U}_1)$, $(\boldsymbol{U}_1, -\boldsymbol{U}_0)$, $(\boldsymbol{U}_0, -\boldsymbol{U}_2)$, $(\boldsymbol{U}_2, -\boldsymbol{U}_0)$, $(\boldsymbol{U}_1, -\boldsymbol{U}_2)$, $(\boldsymbol{U}_2, -\boldsymbol{U}_1)$, $(\boldsymbol{D}, -\boldsymbol{U}_0)$, $(\boldsymbol{U}_0, -\boldsymbol{D})$, $(\boldsymbol{D}, -\boldsymbol{U}_1)$, $(\boldsymbol{U}_1, -\boldsymbol{D})$, $(\boldsymbol{D}, -\boldsymbol{U}_2)$ and $(\boldsymbol{U}_2, -\boldsymbol{D})$. The constructions lead to an interval $[\ell, \mu]$ in one of the spatial variables. If $\ell < \mu$, the clipped line is an extreme edge.

If $\ell = \mu$, the clipped line is an extreme vertex. If $\ell > \mu$, the line is irrelevant in the construction. The projections of the canonical box vertices onto the cylinder axis are quantities that show up in the constructions. Additionally, the projection values show up with an added or subtracted term $h/2$. Specifically,

$$t_i = \boldsymbol{D} \cdot (\boldsymbol{V}_i - \boldsymbol{C}), \;\; s_i^+ = t_i + h/2, \;\; s_i^- = t_i - h/2 \tag{33}$$

The following text blocks describe the algebra for the 12 pairs of normals.

Pair $(\boldsymbol{U}_0, -\boldsymbol{U}_1)$

The planes are $\boldsymbol{U}_0 \cdot \boldsymbol{X} = e_0$ and $-\boldsymbol{U}_1 \cdot \boldsymbol{X} = e_1$. The line of intersection is

$$\boldsymbol{X} = (e_0, -e_1, x_2) \tag{34}$$

with parameter $x_2 \in \mathbb{R}$. The line must be clipped by $|\boldsymbol{D} \cdot (\boldsymbol{X} - \boldsymbol{C})| \le h/2$ and $|\boldsymbol{U}_2 \cdot \boldsymbol{X}| \le e_2$. Solving for x_2, the inequalities are

$$\ell = \max\left\{-e_2, -e_2 - s_1^+/d_2\right\} \le x_2 \le \min\left\{e_2, e_2 - s_5^-/d_2\right\} = \mu \tag{35}$$

The computations are equivalent to clipping the segment $\langle \boldsymbol{V}_1, \boldsymbol{V}_5 \rangle$ by the cylinder slab planes. The endpoints of the segment are

$$\boldsymbol{X}(\ell) = \left\{ \begin{array}{ll} (e_0, -e_1, -e_2), & s_1^+ \ge 0 \\ (e_0, -e_1, -e_2 - s_1^+/d_2), & s_1^+ < 0 \end{array} \right\}, \;\; \boldsymbol{X}(\mu) = \left\{ \begin{array}{ll} (e_0, -e_1, e_2), & s_5^- \le 0 \\ (e_0, -e_1, e_2 - s_5^-/d_2), & s_5^- > 0 \end{array} \right\} \tag{36}$$

Pair $(\boldsymbol{U}_1, -\boldsymbol{U}_0)$

The planes are $\boldsymbol{U}_1 \cdot \boldsymbol{X} = e_1$ and $-\boldsymbol{U}_0 \cdot \boldsymbol{X} = e_0$. The line of intersection is

$$\boldsymbol{X} = (-e_0, e_1, x_2) \tag{37}$$

with parameter $x_2 \in \mathbb{R}$. The line must be clipped by $|\boldsymbol{D} \cdot (\boldsymbol{X} - \boldsymbol{C})| \le h/2$ and $|\boldsymbol{U}_2 \cdot \boldsymbol{X}| \le e_2$. Solving for x_2, the inequalities are

$$\ell = \max\left\{-e_2, -e_2 - s_2^+/d_2\right\} \le x_2 \le \min\left\{e_2, e_2 - s_6^-/d_2\right\} = \mu \tag{38}$$

The computations are equivalent to clipping the segment $\langle \boldsymbol{V}_2, \boldsymbol{V}_6 \rangle$ by the cylinder slab planes. The endpoints of the segment are

$$\boldsymbol{X}(\ell) = \left\{ \begin{array}{ll} (-e_0, e_1, -e_2), & s_2^+ \ge 0 \\ (-e_0, e_1, -e_2 - s_2^+/d_2), & s_2^+ < 0 \end{array} \right\}, \;\; \boldsymbol{X}(\mu) = \left\{ \begin{array}{ll} (-e_0, e_1, e_2), & s_6^- \le 0 \\ (-e_0, e_1, e_2 - s_6^-/d_2), & s_6^- > 0 \end{array} \right\} \tag{39}$$

Pair $(\boldsymbol{U}_0, -\boldsymbol{U}_2)$

The planes are $\boldsymbol{U}_0 \cdot \boldsymbol{X} = e_0$ and $-\boldsymbol{U}_2 \cdot \boldsymbol{X} = e_2$. The line of intersection is

$$\boldsymbol{X} = (e_0, x_1, -e_2) \tag{40}$$

with parameter $x_1 \in \mathbb{R}$. The line must be clipped by $|\boldsymbol{D} \cdot (\boldsymbol{X} - \boldsymbol{C})| \le h/2$ and $|\boldsymbol{U}_1 \cdot \boldsymbol{X}| \le e_1$. Solving for x_1, the inequalities are

$$\ell = \max\left\{-e_1, -e_1 - s_1^+/d_1\right\} \le x_1 \le \min\left\{e_1, e_1 - s_3^-/d_1\right\} = \mu \tag{41}$$

The computations are equivalent to clipping the segment $\langle \boldsymbol{V}_1, \boldsymbol{V}_3 \rangle$ by the cylinder slab planes. The endpoints of the segment are

$$\boldsymbol{X}(\ell) = \left\{ \begin{array}{ll} (e_0, -e_1, -e_2), & s_1^+ \ge 0 \\ (e_0, -e_1 - s_1^+/d_1, -e_2), & s_1^+ < 0 \end{array} \right\}, \quad \boldsymbol{X}(\mu) = \left\{ \begin{array}{ll} (e_0, e_1, -e_2), & s_3^- \le 0 \\ (e_0, e_1 - s_3^-/d_1, -e_2), & s_3^- > 0 \end{array} \right\} \tag{42}$$

Pair $(\boldsymbol{U}_2, -\boldsymbol{U}_0)$

The planes are $\boldsymbol{U}_2 \cdot \boldsymbol{X} = e_2$ and $-\boldsymbol{U}_0 \cdot \boldsymbol{X} = e_0$. The line of intersection is

$$\boldsymbol{X} = (-e_0, x_1, e_2) \tag{43}$$

with parameter $x_1 \in \mathbb{R}$. The line must be clipped by $|\boldsymbol{D} \cdot (\boldsymbol{X} - \boldsymbol{C})| \le h/2$ and $|\boldsymbol{U}_1 \cdot \boldsymbol{X}| \le e_1$. Solving for x_1, the inequalities are

$$\ell = \max\left\{-e_1, -e_1 - s_4^+/d_1\right\} \le x_1 \le \min\left\{e_1, e_1 - s_6^-/d_1\right\} = \mu \tag{44}$$

The computations are equivalent to clipping the segment $\langle \boldsymbol{V}_4, \boldsymbol{V}_6 \rangle$ by the cylinder slab planes. The endpoints of the segment are

$$\boldsymbol{X}(\ell) = \left\{ \begin{array}{ll} (-e_0, -e_1, e_2), & s_4^+ \ge 0 \\ (-e_0, -e_1 - s_4^+/d_1, e_2), & s_4^+ < 0 \end{array} \right\}, \quad \boldsymbol{X}(\mu) = \left\{ \begin{array}{ll} (-e_0, e_1, e_2), & s_6^- \le 0 \\ (-e_0, e_1 - s_6^-/d_1, e_2), & s_6^- > 0 \end{array} \right\} \tag{45}$$

Pair $(\boldsymbol{U}_1, -\boldsymbol{U}_2)$

The planes are $\boldsymbol{U}_1 \cdot \boldsymbol{X} = e_1$ and $-\boldsymbol{U}_2 \cdot \boldsymbol{X} = e_2$. The line of intersection is

$$\boldsymbol{X} = (x_0, e_1, -e_2) \tag{46}$$

with parameter $x_0 \in \mathbb{R}$. The line must be clipped by $|\boldsymbol{D} \cdot (\boldsymbol{X} - \boldsymbol{C})| \le h/2$ and $|\boldsymbol{U}_0 \cdot \boldsymbol{X}| \le e_0$. Solving for x_0, the inequalities are

$$\ell = \max\left\{-e_0, -e_0 - s_2^+/d_0\right\} \le x_0 \le \min\left\{e_0, e_0 - s_3^-/d_0\right\} = \mu \tag{47}$$

The computations are equivalent to clipping the segment $\langle \boldsymbol{V}_2, \boldsymbol{V}_3 \rangle$ by the cylinder slab planes. The endpoints of the segment are

$$\boldsymbol{X}(\ell) = \left\{ \begin{array}{ll} (-e_0, e_1, -e_2), & s_2^+ \ge 0 \\ (-e_0 - s_2^+/d_0, e_1, -e_2), & s_2^+ < 0 \end{array} \right\}, \quad \boldsymbol{X}(\mu) = \left\{ \begin{array}{ll} (e_0, e_1, -e_2), & s_3^- \le 0 \\ (e_0 - s_3^-/d_0, e_1, -e_2), & s_3^- > 0 \end{array} \right\} \tag{48}$$

Pair $(\boldsymbol{U}_2, -\boldsymbol{U}_1)$

The planes are $\boldsymbol{U}_2 \cdot \boldsymbol{X} = e_2$ and $-\boldsymbol{U}_1 \cdot \boldsymbol{X} = e_1$. The line of intersection is

$$\boldsymbol{X} = (x_0, -e_1, e_2) \tag{49}$$

with parameter $x_0 \in \mathbb{R}$. The line must be clipped by $|\boldsymbol{D} \cdot (\boldsymbol{X} - \boldsymbol{C})| \le h/2$ and $|\boldsymbol{U}_0 \cdot \boldsymbol{X}| \le e_0$. Solving for x_0, the inequalities are

$$\ell = \max\left\{-e_0, -e_0 - s_4^+/d_0\right\} \le x_0 \le \min\left\{e_0, e_0 - s_5^-/d_0\right\} = \mu \tag{50}$$

The computations are equivalent to clipping the segment $\langle \boldsymbol{V}_4, \boldsymbol{V}_5 \rangle$ by the cylinder slab planes. The endpoints of the segment are

$$\boldsymbol{X}(\ell) = \left\{ \begin{array}{ll} (-e_0, -e_1, e_2), & s_4^+ \ge 0 \\ (-e_0 - s_4^+/d_0, -e_1, e_2), & s_4^+ < 0 \end{array} \right\}, \quad \boldsymbol{X}(\mu) = \left\{ \begin{array}{ll} (e_0, -e_1, e_2), & s_5^- \le 0 \\ (e_0 - s_5^-/d_0, -e_1, e_2), & s_5^- > 0 \end{array} \right\} \tag{51}$$

Pair $(\boldsymbol{U}_0, -\boldsymbol{D})$

The planes are $\boldsymbol{U}_0 \cdot \boldsymbol{X} = e_0$ and $-\boldsymbol{D} \cdot (\boldsymbol{X} - \boldsymbol{C}) = h/2$. The line of intersection is

$$\boldsymbol{X} = (e_0, (-h/2 - d_0(e_0 - c_0) - d_1(-c_1) - d_2(x_2 - c_2))/d_1, x_2) \tag{52}$$

with parameter $x_2 \in \mathbb{R}$. The line must be clipped by $|\boldsymbol{U}_1 \cdot \boldsymbol{X}| \le e_1$ and $|\boldsymbol{U}_2 \cdot \boldsymbol{X}| \le e_2$. Solving for x_2, the inequalities are

$$\ell = \max\left\{-e_2, -e_2 - s_3^+/d_2\right\} \le x_2 \le \min\left\{e_2, e_2 - s_5^+/d_2\right\} = \mu \tag{53}$$

The endpoints of the segment are

$$\boldsymbol{X}(\ell) = \left\{ \begin{array}{ll} (e_0, e_1 - s_3^+/d_1, -e_2), & s_3^+ \ge 0 \\ (e_0, e_1, -e_2 - s_3^+/d_2), & s_3^+ < 0 \end{array} \right\}, \quad \boldsymbol{X}(\mu) = \left\{ \begin{array}{ll} (e_0, -e_1 - s_5^+/d_1, e_2), & s_5^+ \le 0 \\ (e_0, -e_1, e_2 - s_5^+/d_2), & s_5^+ > 0 \end{array} \right\} \tag{54}$$

Pair $(\boldsymbol{D}, -\boldsymbol{U}_0)$

The planes are $\boldsymbol{D} \cdot (\boldsymbol{X} - \boldsymbol{C}) = h/2$ and $-\boldsymbol{U}_0 \cdot \boldsymbol{X} = e_0$. The line of intersection is

$$\boldsymbol{X} = (-e_0, (h/2 - d_0(-e_0 - c_0) - d_1(-c_1) - d_2(x_2 - c_2))/d_1, x_2) \tag{55}$$

with parameter $x_2 \in \mathbb{R}$. The line must be clipped by $|\boldsymbol{U}_1 \cdot \boldsymbol{X}| \le e_1$ and $|\boldsymbol{U}_2 \cdot \boldsymbol{X}| \le e_2$. Solving for x_2, the inequalities are

$$\ell = \max\left\{-e_2, -e_2 - s_2^-/d_2\right\} \le x_2 \le \min\left\{e_2, e_2 - s_4^-/d_2\right\} = \mu \tag{56}$$

The endpoints of the segment are

$$\boldsymbol{X}(\ell) = \left\{ \begin{array}{ll} (-e_0, e_1 - s_2^-/d_1, -e_2), & s_2^- \ge 0 \\ (-e_0, e_1, -e_2 - s_2^-/d_2), & s_2^- < 0 \end{array} \right\}, \quad \boldsymbol{X}(\mu) = \left\{ \begin{array}{ll} (-e_0, -e_1 - s_4^-/d_1, e_2), & s_4^- \le 0 \\ (-e_0, -e_1, e_2 - s_4^-/d_1), & s_4^- > 0 \end{array} \right\} \tag{57}$$

Pair $(\boldsymbol{U}_1, -\boldsymbol{D})$

The planes are $\boldsymbol{U}_1 \cdot \boldsymbol{X} = e_1$ and $-\boldsymbol{D} \cdot (\boldsymbol{X} - \boldsymbol{C}) = h/2$. The line of intersection is

$$\boldsymbol{X} = (x_0, e_1, (-h/2 - d_0(x_0 - c_0) - d_1(e_1 - c_1) - d_2(-c_2))/d_2) \tag{58}$$

with parameter $x_0 \in \mathbb{R}$. The line must be clipped by $|\boldsymbol{U}_0 \cdot \boldsymbol{X}| \le e_0$ and $|\boldsymbol{U}_2 \cdot \boldsymbol{X}| \le e_2$. Solving for x_0, the inequalities are

$$\ell = \max\left\{-e_0, -e_0 - s_6^+/d_0\right\} \le x_2 \le \min\left\{e_0, e_0 - s_3^+/d_0\right\} = \mu \tag{59}$$

The endpoints of the segment are

$$\boldsymbol{X}(\ell) = \left\{ \begin{array}{ll} (-e_0, e_1, e_2 - s_6^+/d_2), & s_6^+ \ge 0 \\ (-e_0 - s_6^+/d_0, e_1, e_2), & s_6^+ < 0 \end{array} \right\}, \quad \boldsymbol{X}(\mu) = \left\{ \begin{array}{ll} (e_0, e_1, -e_2 - s_3^+/d_2), & s_3^+ \le 0 \\ (e_0 - s_3^+/d_0, e_1, -e_2), & s_3^+ > 0 \end{array} \right\} \tag{60}$$

Pair $(\boldsymbol{D}, -\boldsymbol{U}_1)$

The planes are $\boldsymbol{D} \cdot (\boldsymbol{X} - \boldsymbol{C}) = h/2$ and $\boldsymbol{U}_1 \cdot \boldsymbol{X} = -e_1$ The line of intersection is

$$\boldsymbol{X} = (x_0, -e_1, (h/2 - d_0(x_0 - c_0) - d_1(-e_1 - c_1) - d_2(-c_2))/d_2) \tag{61}$$

with parameter $x_0 \in \mathbb{R}$. The line must be clipped by $|\boldsymbol{U}_0 \cdot \boldsymbol{X}| \le e_0$ and $|\boldsymbol{U}_2 \cdot \boldsymbol{X}| \le e_2$. Solving for x_0, the inequalities are

$$\ell = \max\left\{-e_0, -e_0 - s_4^-/d_0\right\} \le x_2 \le \min\left\{e_0, e_0 - s_1^-/d_0\right\} = \mu \tag{62}$$

The endpoints of the segment are

$$\boldsymbol{X}(\ell) = \left\{ \begin{array}{ll} (-e_0, -e_1, e_2 - s_4^-/d_2), & s_4^- \ge 0 \\ (-e_0 - s_4^-/d_0, -e_1, e_2), & s_4^- < 0 \end{array} \right\}, \quad \boldsymbol{X}(\mu) = \left\{ \begin{array}{ll} (e_0, -e_1, -e_2 - s_1^-/d_2), & s_1^- \le 0 \\ (e_0 - s_1^-/d_0, -e_1, -e_2), & s_1^- > 0 \end{array} \right\} \tag{63}$$

Pair $(\boldsymbol{U}_2, -\boldsymbol{D})$

The planes are $\boldsymbol{U}_2 \cdot \boldsymbol{X} = e_2$ and $-\boldsymbol{D} \cdot (\boldsymbol{X} - \boldsymbol{C}) = h/2$. The line of intersection is

$$\boldsymbol{X} = ((-h/2 - d_0(-c_0) - d_1(x_1 - c_1) - d_2(e_2 - c_2))/d_0, x_1, e_2) \tag{64}$$

with parameter $x_1 \in \mathbb{R}$. The line must be clipped by $|\boldsymbol{U}_0 \cdot \boldsymbol{X}| \le e_0$ and $|\boldsymbol{U}_1 \cdot \boldsymbol{X}| \le e_1$. Solving for x_0, the inequalities are

$$\ell = \max\left\{-e_1, -e_1 - s_5^+/d_1\right\} \le x_2 \le \min\left\{e_1, e_1 - s_6^+/d_1\right\} = \mu \tag{65}$$

The endpoints of the segment are

$$\boldsymbol{X}(\ell) = \left\{ \begin{array}{ll} (e_0 - s_5^+/d_0, -e_1, e_2), & s_5^+ \ge 0 \\ (e_0, -e_1 - s_5^+/d_1, e_2), & s_5^+ < 0 \end{array} \right\}, \quad \boldsymbol{X}(\mu) = \left\{ \begin{array}{ll} (-e_0 - s_6^+/d_0, e_1, e_2), & s_6^+ \le 0 \\ (-e_0, e_1 - s_6^+/d_1, e_2), & s_6^+ > 0 \end{array} \right\} \tag{66}$$

Pair $(\boldsymbol{D}, -\boldsymbol{U}_2)$

The planes are $\boldsymbol{D} \cdot (\boldsymbol{X} - \boldsymbol{C}) = h/2$ and $\boldsymbol{U}_2 \cdot \boldsymbol{X} = -e_2$ .The line of intersection is

$$\boldsymbol{X} = ((h/2 - d_0(-c_0) - d_1(x_1 - c_1) - d_2(-e_2 - c_2))/d_0, x_1, -e_2) \tag{67}$$

with parameter $x_1 \in \mathbb{R}$. The line must be clipped by $|\boldsymbol{U}_0 \cdot \boldsymbol{X}| \le e_0$ and $|\boldsymbol{U}_1 \cdot \boldsymbol{X}| \le e_1$. Solving for x_0, the inequalities are

$$\ell = \max\left\{-e_1, -e_1 - s_1^-/d_1\right\} \le x_2 \le \min\left\{e_1, e_1 - s_2^-/d_1\right\} = \mu \tag{68}$$

The endpoints of the segment are

$$\boldsymbol{X}(\ell) = \left\{ \begin{array}{ll} (e_0 - s_1^-/d_0, -e_1, -e_2), & s_1^- \ge 0 \\ (e_0, -e_1 - s_1^-/d_1, -e_2), & s_1^- < 0 \end{array} \right\}, \quad \boldsymbol{X}(\mu) = \left\{ \begin{array}{ll} (-e_0 - s_2^-/d_0, e_1, -e_2), & s_2^- \le 0 \\ (-e_0, e_1 - s_2^-/d_1, -e_2), & s_2^- > 0 \end{array} \right\} \tag{69}$$

4.3.3 Test Intersection via Projection and Distance

The 12 pairs of normals are processed. For each corresponding interval $[\ell, \mu]$ that is valid ($\ell \le \mu$), the extreme edge or vertex is projected to the plane containing $\boldsymbol{C}$ and having normal $\boldsymbol{D}$. The squared distance δ^2 from $\boldsymbol{C}$ to the projected edge or vertex is computed and compared to the squared radius r^2. The finite cylinder and canonical box intersect when $\delta^2 \le r^2$. When an extreme edge leads to an intersection, there might be additional extreme edges that were not yet visited. The early exit is not part of the generic convex hull approach where you must compute the entire set of polygon edges before you can start the distance computations from a point to an edge.

Listing 6 contains pseudocode for the test-intersection query when $\boldsymbol{D}$ has no zero-valued components.

Listing 6. Pseudocode to determine whether the box and cylinder intersect when the direction has no zero-valued components.

```
// Robustly compute an orthonormal basis containing v0. The input v0 is unit length. The outputs are v1 and v2.
void ComputeOrthonormalBasis(Vector3 v0, Vector3& v1, Vector3& v2)
{
    if (abs(v0[0]) > abs(v0[1]))
    {
        v1 = { -v0[2], 0, +v0[0] };
    }
    else
    {
        v1 = { 0, +v0[2], -v0[1] };
    }
    Normalize(v1);
    v2 = Cross(v0, v1);
}

// Compute the distance from (0,0) to the projection of the segment <P0,P1>. The projection plane has origin C and is
// spanned by the orthonormal vectors W0 and W1.
Real ComputeSqrDistance(Vector3 P0, Vector3 P1, Vector3 C, Vector3 W0, Vector3 W1)
{
    Vector3 P0mC = P0 - C;
    Vector3 P1mC = P1 - C;
    Vector2 Q0{ Dot(W0, P0mC), Dot(W1, P0mC) };
    Vector2 Q1{ Dot(W0, P1mC), Dot(W1, P1mC) };
```

```
    Vector2 direction = Q1 - Q0;
    Real s = Dot(direction, Q1);
    if (s <= 0)
    {
        return Dot(Q1, Q1);
    }
    else
    {
        s = Dot(direction, Q0);
        if (s >= 0)
        {
            return Dot(Q0, Q0);
        }
        else
        {
            s /= Dot(direction, direction);
            Vector2 closest = Q0 + s * direction;
            return Dot(closest, closest);
        }
    }
}

bool Intersects0(Vector3 C, Vector3 D, Real r, Real hDiv2, Vector3 E)
{
    // Test whether the cylinder axis and canonical box intersect.
    array<Real, 3> negEmCDivD = {(-E[0] - C[0]) / D[0], (-E[1] - C[1]) / D[1], (-E[2] - C[2]) / D[2] };
    array<Real, 3> posEmCDivD = {(+E[0] - C[0]) / D[0], (+E[1] - C[1]) / D[1], (+E[2] - C[2]) / D[2] };
    Real lower = max(max(negEmCDivD[0], negEmCDivD[1]), max(negEmCDivD[2], -hDiv2));
    Real upper = min(min(posEmCDivD[0], posEmCDivD[1]), min(posEmCDivD[2], +hDiv2));
    if (lower <= upper)
    {
        return true;
    }

    Real sqrRadius = r * r;

    // Compute t_i = D . (V_i - C) for box vertices V_i. These are used in computing the intervals associated
    // with extreme edges.
    Real dotDC = Dot(D, C);
    Real d0e0 = D[0] * E[0], d1e1 = D[1] * E[1], d2e2 = D[2] * E[2];
    Real t1 = +d0e0 - d1e1 - d2e2 - dotDC, s1p = t1 + hDiv2, s1n = t1 - hDiv2;
    Real t2 = -d0e0 + d1e1 - d2e2 - dotDC, s2p = t2 + hDiv2, s2n = t2 - hDiv2;
    Real t3 = +d0e0 + d1e1 - d2e2 - dotDC, s3p = t3 + hDiv2, s3n = t3 - hDiv2;
    Real t4 = -d0e0 - d1e1 + d2e2 - dotDC, s4p = t4 + hDiv2, s4n = t4 - hDiv2;
    Real t5 = +d0e0 - d1e1 + d2e2 - dotDC, s5p = t5 + hDiv2, s5n = t5 - hDiv2;
    Real t6 = -d0e0 + d1e1 + d2e2 - dotDC, s6p = t6 + hDiv2, s6n = t6 - hDiv2;

    // Compute an orthonormal basis containing D.
    Vector3 W0, W1;
    ComputeOrthonormalBasis(D, W0, W1);

    Real lower, upper, sqrDistance;
    Vector3 P0, P1;

    // (U_0, -U_1)
    lower = (s1p >= 0 ? -E[2] : -E[2] - s1p / D[2]);
    upper = (s5n <= 0 ? +E[2] : +E[2] - s5n / D[2]);
    if (lower <= upper)
    {
        P0 = { +E[0], -E[1], lower };
        P1 = { +E[0], -E[1], upper };
        sqrDistance = ComputeSqrDistance(P0, P1, C, W0, W1);
        if (sqrDistance <= sqrRadius)
        {
            return true;
        }
    }

    // (U_1, -U_0)
```

```
lower = (s2p >= 0 ? -E[2] : -E[2] - s2p / D[2]);
upper = (s6n <= 0 ? +E[2] : +E[2] - s6n / D[2]);
if (lower <= upper)
{
    P0 = { -E[0], +E[1], lower };
    P1 = { -E[0], +E[1], upper };
    sqrDistance = ComputeSqrDistance(P0, P1, C, W0, W1);
    if (sqrDistance <= sqrRadius)
    {
        return true;
    }
}

// (U0, -U2)
lower = (s1p >= 0 ? -E[1] : -E[1] - s1p / D[1]);
upper = (s3n <= 0 ? +E[1] : +E[1] - s3n / D[1]);
if (lower <= upper)
{
    P0 = { +E[0], lower, -E[2] };
    P1 = { +E[0], upper, -E[2] };
    sqrDistance = ComputeSqrDistance(P0, P1, C, W0, W1);
    if (sqrDistance <= sqrRadius)
    {
        return true;
    }
}

// (U2, -U0)
lower = (s4p >= 0 ? -E[1] : -E[1] - s4p / D[1]);
upper = (s6n <= 0 ? +E[1] : +E[1] - s6n / D[1]);
if (lower <= upper)
{
    P0 = { -E[0], lower, +E[2] };
    P1 = { -E[0], upper, +E[2] };
    sqrDistance = ComputeSqrDistance(P0, P1, C, W0, W1);
    if (sqrDistance <= sqrRadius)
    {
        return true;
    }
}

// (U1, -U2)
lower = (s2p >= 0 ? -E[0] : -E[0] - s2p / D[0]);
upper = (s3n <= 0 ? +E[0] : +E[0] - s3n / D[0]);
if (lower <= upper)
{
    P0 = { lower, +E[1], -E[2] };
    P1 = { upper, +E[1], -E[2] };
    sqrDistance = ComputeSqrDistance(P0, P1, C, W0, W1);
    if (sqrDistance <= sqrRadius)
    {
        return true;
    }
}

// (U2, -U1)
lower = (s4p >= 0 ? -E[0] : -E[0] - s4p / D[0]);
upper = (s5n <= 0 ? +E[0] : +E[0] - s5n / D[0]);
if (lower <= upper)
{
    P0 = { lower, -E[1], +E[2] };
    P1 = { upper, -E[1], +E[2] };
    sqrDistance = ComputeSqrDistance(P0, P1, C, W0, W1);
    if (sqrDistance <= sqrRadius)
    {
        return true;
    }
}

// (U0, -D)
lower = (s3p >= 0 ? -E[2] : -E[2] - s3p / D[2]);
```

```
upper = (s5p <= 0 ? +E[2] : +E[2] - s5p / D[2]);
if (lower <= upper)
{
    if (s3p >= 0)
    {
        P0 = { +E[0], +E[1] - s3p / D[1], -E[2] };
    }
    else
    {
        P0 = { +E[0], +E[1], -E[2] - s3p / D[2] };
    }

    if (s5p <= 0)
    {
        P1 = { +E[0], -E[1] - s5p / D[1], +E[2] };
    }
    else
    {
        P1 = { +E[0], -E[1], +E[2] - s5p / D[2] };
    }

    sqrDistance = ComputeSqrDistance(P0, P1, C, W0, W1);
    if (sqrDistance <= sqrRadius)
    {
        return true;
    }
}

// (D, -U0)
lower = (s2n >= 0 ? -E[2] : -E[2] - s2n / D[2]);
upper = (s4n <= 0 ? +E[2] : +E[2] - s4n / D[2]);
if (lower <= upper)
{
    if (s2n >= 0)
    {
        P0 = { -E[0], +E[1] - s2n / D[1], -E[2] };
    }
    else
    {
        P0 = { -E[0], +E[1], -E[2] - s2n / D[2] };
    }

    if (s4n <= 0)
    {
        P1 = { -E[0], -E[1] - s4n / D[1], +E[2] };
    }
    else
    {
        P1 = { -E[0], -E[1], +E[2] - s4n / D[2] };
    }

    sqrDistance = ComputeSqrDistance(P0, P1, C, W0, W1);
    if (sqrDistance <= sqrRadius)
    {
        return true;
    }
}

// (U1, -D)
lower = (s6p >= 0 ? -E[0] : -E[0] - s6p / D[0]);
upper = (s3p <= 0 ? +E[0] : +E[0] - s3p / D[0]);
if (lower <= upper)
{
    if (s6p >= 0)
    {
        P0 = { -E[0], +E[1], +E[2] - s6p / D[2] };
    }
    else
    {
        P0 = { -E[0] - s6p / D[0], +E[1], +E[2] };
    }
```

```
    if (s3p <= 0)
    {
        P1 = { +E[0], +E[1], -E[2] - s3p / D[2] };
    }
    else
    {
        P1 = { +E[0] - s3p / D[0], -E[1], -E[2] };
    }

    sqrDistance = ComputeSqrDistance(P0, P1, C, W0, W1);
    if (sqrDistance <= sqrRadius)
    {
        return true;
    }
}

// (D, -U1)
lower = (s4n >= 0 ? -E[0] : -E[0] - s4n / D[0]);
upper = (s1n <= 0 ? +E[0] : +E[0] - s1n / D[0]);
if (lower <= upper)
{
    if (s4n >= 0)
    {
        P0 = { -E[0], -E[1], +E[2] - s4n / D[2] };
    }
    else
    {
        P0 = { -E[0] - s4n / D[0], -E[1], +E[2] };
    }

    if (s1n <= 0)
    {
        P1 = { +E[0], -E[1], -E[2] - s1n / D[2] };
    }
    else
    {
        P1 = { +E[0] - s1n / D[0], -E[1], -E[2] };
    }

    sqrDistance = ComputeSqrDistance(P0, P1, C, W0, W1);
    if (sqrDistance <= sqrRadius)
    {
        return true;
    }
}

// (U2, -D)
lower = (s5p >= 0 ? -E[1] : -E[1] - s5p / D[2]);
upper = (s6p <= 0 ? +E[1] : +E[1] - s6p / D[2]);
if (lower <= upper)
{
    if (s5p >= 0)
    {
        P0 = { +E[0] - s5p / D[0], -E[1], +E[2] };
    }
    else
    {
        P0 = { +E[0], -E[1] - s5p / D[1], +E[2] };
    }

    if (s6p <= 0)
    {
        P1 = { -E[0] - s6p / D[0], +E[1], +E[2] };
    }
    else
    {
        P1 = { -E[0], E[1] - s6p / D[1], +E[2] };
    }

    sqrDistance = ComputeSqrDistance(P0, P1, C, W0, W1);
```

```
        if (sqrDistance <= sqrRadius)
        {
            return true;
        }
    }

    // (D, -U2)
    lower = (s1n >= 0 ? -E[1] : -E[1] - s1n / D[2]);
    upper = (s2n <= 0 ? +E[1] : +E[1] - s2n / D[2]);
    if (lower <= upper)
    {
        if (s1n >= 0)
        {
            P0 = { +E[0] - s1n / D[0], -E[1], -E[2] };
        }
        else
        {
            P0 = { +E[0], -E[1] - s1n / D[1], -E[2] };
        }

        if (s2n <= 0)
        {
            P1 = { -E[0] - s2n / D[0], +E[1], -E[2] };
        }
        else
        {
            P1 = { -E[0], E[1] - s2n / D[1], -E[2] };
        }

        sqrDistance = ComputeSqrDistance(P0, P1, C, W0, W1);
        if (sqrDistance <= sqrRadius)
        {
            return true;
        }
    }

    return false;
}
```

4.4 The Top-Level Call of the Query

Listing 7 contains pseudocode for the test-intersection query between a finite cylinder and a canonical box.

Listing 7. Pseudocode for the test-intersection query between a finite cylinder and a canonical box.

```
// The input cylinder must have finite height. The return value is true if and only if the box and cylinder intersect.
bool TestIntersectionQuery(CanonicalBox3 box, Cylinder3 cylinder)
{
    if (BoxIsOutsideCylinderSlab(box, cylinder))
    {
        // The box does not intersect the slab, so it does not intersect the cylinder.
        return false;
    }

    // Apply reflections to obtain a cylinder whose axis direction is in the first octant (positive- or zero-valued
    // components). The reflections applied to the canonical box do not require any computational changes.
    Vector3 C = cylinder.center;
    Vector3 D = cylinder.direction;
    Real r = cylinder.radius;
    Real hDiv2 = cylinder.height / 2;
    Vector3 E = box.extent;
    for (int i = 0; i < 3; ++i)
```

```
    {
        if (D[i] < 0)
        {
            C[i] = -C[i];
            D[i] = -D[i];
        }
    }

    // D is now in the first octant. The box vertices are
    // V[0] = (-E[0],-E[1],-E[2]), V[4] = (-E[0],-E[1],+E[2])
    // V[1] = (+E[0],-E[1],-E[2]), V[5] = (+E[0],-E[1],+E[2])
    // V[2] = (-E[0],+E[1],-E[2]), V[6] = (-E[0],+E[1],+E[2])
    // V[3] = (+E[0],+E[1],-E[2]), V[7] = (+E[0],+E[1],+E[2])
    if (D[0] > 0)
    {
        if (D[1] > 0)
        {
            if (D[2] > 0)  // (+,+,+)
            {
                return Intersects0(C, D, r, hDiv2, E);
            }
            else  // (+,+,0)
            {
                return Intersects1({ 0, 1, 2 }, C, D, r, hDiv2, E);
            }
        }
        else
        {
            if (D[2] > 0)  // (+,0,+)
            {
                return Intersects1({ 2, 0, 1 }, C, D, r, hDiv2, E);
            }
            else  // (+,0,0)
            {
                return Intersects2({ 0, 1, 2 }, C, r, E);
            }
        }
    }
    else
    {
        if (D[1] > 0)
        {
            if (D[2] > 0)  // (0,+,+)
            {
                return Intersects1({ 1, 2, 0 }, C, D, r, hDiv2, E);
            }
            else  // (0,+,0)
            {
                return Intersects2({ 1, 2, 0 }, C, r, E);
            }
        }
        else
        {
            if (D[2] > 0)  // (0,0,+)
            {
                return Intersects2({ 2, 0, 1 }, C, r, E);
            }
            else  // (0,0,0), which cannot occur
            {
                return false;
            }
        }
    }
}
```

References

[1] Dave Eberly. *Robust and Error-Free Geometric Computing*. CRC Press, Taylor & Francis Group LLC, Boca Raton, FL, 2020.

[2] Wikipedia. Convex hull algorithms.
https://en.wikipedia.org/wiki/Convex_hull_algorithms.
accessed December 12, 2021.

[3] Wikipedia. Fourier–Motzkin elimination.
https://en.wikipedia.org/wiki/Fourier-Motzkin_elimination.
accessed December 8, 2021.

[4] Wolfram Research, Inc. *Mathematica 12.3.1*. Wolfram Research, Inc., Champaign, Illinois, 2021.