Robust Intersection of Ellipses

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

Created: June 16, 2023

Contents

1 Introduction

This document describes how to compute the points of intersection of two ellipses in a robust manner, using floating-point arithmetic or rational arithmetic. When using rational arithmetic, it is not always possible to compute the points exactly. The algorithm involves computing roots to quartic polynomials, which are not generally representable as rational numbers.

One goal is to have a robust estimator for the quartic roots in order to minimize the rounding errors that occur in the root finder. Another goal is to ensure that floating-point computations have a minimum amount of rounding error. This goal is accomplished by using fused-multiply-add operations for expressions of the form $xy - zw$. Using only standard floating-point multiplication and subtraction, subtractive cancellation of significant bits can occur when xy and zw are the same sign and nearly the same magnitude.

2 Ellipse Representations

Ellipses can be represented algebraically using two formulations: the *standard form* and the *quadratic equation form*. The standard form requires specifying unit-length ellipse axis directions, but when using floating-point arithmetic, the normalization of vectors to obtain unit-length directions introduces rounding errors. The setup for the standard form can be written instead to use non-unit-length ellipse axis directions, which supports a rational formulation of the ellipse. The standard form can be converted to a quadratic equation in two variables. The ellipse is represented as the zero-valued level set of the quadratic equation.

2.1 The Standard Form for an Ellipse

Let the *ellipse center* be $\boldsymbol{C}$. Let the *ellipse axis directions* be $\boldsymbol{U}$ and $\boldsymbol{V} = -\boldsymbol{U}^{\perp}$, where $\boldsymbol{U}$ is a unit-length vector and where $(x, y)^{\perp} = (y, -x)$. The vector $\boldsymbol{V}$ is unit length, perpendicular to $\boldsymbol{U}$, and is a counterclockwise rotation of $\boldsymbol{U}$ in the xy-plane. Let the *ellipse extents* be $a > 0$ and $b > 0$. The extent a is the distance from $\boldsymbol{C}$ to extreme points $\boldsymbol{C} \pm a\boldsymbol{U}$ and the extent b is the distance from $\boldsymbol{C}$ to extreme points $\boldsymbol{C} \pm b\boldsymbol{V}$. If $a > b$, the line through $\boldsymbol{C}$ in the direction $\boldsymbol{U}$ is referred to as the major axis of the ellipse. The line through $\boldsymbol{C}$ in the direction $\boldsymbol{V}$ is referred to as the minor axis of the ellipse.

An ellipse point $\boldsymbol{X} = (x_0, x_1)$ is represented in the coordinate system $\{\boldsymbol{C}; \boldsymbol{U}, \boldsymbol{V}\}$ by $\boldsymbol{X} = \boldsymbol{C} + \mu\boldsymbol{U} + \nu\boldsymbol{V}$, where $(\mu/a)^2 + (\nu/b)^2 = 1$. We can compute the coordinates by projection, $\mu = \boldsymbol{U} \cdot (\boldsymbol{X} - \boldsymbol{C})$ and $\nu = \boldsymbol{V} \cdot (\boldsymbol{X} - \boldsymbol{C})$, in which case

$$\begin{aligned} 1 &= (\mu/a)^2 + (\nu/b)^2 \\ &= (\boldsymbol{U} \cdot (\boldsymbol{X} - \boldsymbol{C})/a)^2 + (\boldsymbol{V} \cdot (\boldsymbol{X} - \boldsymbol{C})/b)^2 \\ &= (\boldsymbol{X} - \boldsymbol{C})^{\mathsf{T}} \left(\boldsymbol{U}\boldsymbol{U}^{\mathsf{T}}/a^2 + \boldsymbol{V}\boldsymbol{V}^{\mathsf{T}}/b^2 \right) (\boldsymbol{X} - \boldsymbol{C}) \\ &= (\boldsymbol{X} - \boldsymbol{C})^{\mathsf{T}} M (\boldsymbol{X} - \boldsymbol{C}) \end{aligned} \tag{1}$$

where the last equality defines the positive definite matrix $M = [m_{ij}]$ where i is the row index with $0 \le i < 2$ and j is the column index with $0 \le j < 2$. By definition, M is symmetric, has positive eigenvalues, and is invertible. The *standard form* of the ellipse is the equation $(\boldsymbol{X} - \boldsymbol{C})^{\mathsf{T}} M (\boldsymbol{X} - \boldsymbol{C}) = 1$.

Equation (1) assumes unit-length vectors $\boldsymbol{U}$ and $\boldsymbol{V} = -\boldsymbol{U}^{\perp}$. When computing with floating-point arithmetic, normalization of a vector $\widehat{\boldsymbol{U}}$ is typically used to generate $\boldsymbol{U} = \widehat{\boldsymbol{U}}/|\widehat{\boldsymbol{U}}|$. The normalization involves computing

a sum of squares, which has rounding errors, followed by a square root, which has rounding errors, and then followed by a division which introduces yet more rounding errors.

The normalization can be avoided, supporting a rational representation of an ellipse. The ellipse parameters are specified as finite floating-point numbers, which are rational numbers. Let $\widehat{\boldsymbol{U}}$ be an ellipse axis direction that is not unit length. Choose $\widehat{\boldsymbol{V}} = -\widehat{\boldsymbol{U}}^{\perp}$, in which case $|\widehat{\boldsymbol{U}}|^2 = |\widehat{\boldsymbol{V}}|^2$ and the common value is represented by a rational number. The matrix M of equation (1) is

$$M = \frac{1}{|\widehat{\boldsymbol{U}}|^2}\left(\frac{\widehat{\boldsymbol{U}}\widehat{\boldsymbol{U}}^{\mathsf{T}}}{a^2} + \frac{\widehat{\boldsymbol{V}}\widehat{\boldsymbol{V}}^{\mathsf{T}}}{b^2}\right) = \begin{bmatrix} m_{00} & m_{01} \\ m_{01} & m_{11} \end{bmatrix} \tag{2}$$

with determinant $\det(M) = m_{00}m_{11} - m_{01}^2$, and the inverse M^{-1} is

$$M^{-1} = \frac{1}{|\widehat{\boldsymbol{U}}|^2}\left(a^2\widehat{\boldsymbol{U}}\widehat{\boldsymbol{U}}^{\mathsf{T}} + b^2\widehat{\boldsymbol{V}}\widehat{\boldsymbol{V}}^{\mathsf{T}}\right) = \frac{1}{\det(M)}\begin{bmatrix} m_{11} & -m_{01} \\ -m_{01} & m_{00} \end{bmatrix} \tag{3}$$

2.2 The Quadratic Equation Form for an Ellipse

Equation (1) can be expanded to a quadratic equation. Let $\boldsymbol{X} = (x_0, x_1)$, $\boldsymbol{C} = (c_0, c_1)$, and $M = [m_{ij}]$ for $0 \le i < 2$ and $0 \le j < 2$. The expansion is

$$Q(x_0, x_1) = q_0 + q_1x_0 + q_2y_0 + q_3x_0^2 + q_4x_0x_1 + q_5x_1^2 = 0 \tag{4}$$

where

$$\begin{aligned} &q_0 = m_{00}c_0^2 + 2m_{01}c_0c_1 + m_{11}c_1^2 - 1, \\ &q_1 = -2(m_{00}c_0 + m_{01}c_1), \;\; q_2 = -2(m_{01}c_0 + m_{11}c_1), \\ &q_3 = m_{00}, \;\; q_4 = 2m_{01}, \;\; q_5 = m_{11} \end{aligned} \tag{5}$$

The positive definiteness of M implies $q_3 = m_{00} > 0$, $q_5 = m_{11} > 0$, and $q_3q_5 - q_4^2/4 = m_{00}m_{11} - m_{01}^2 > 0$. These inequalities are expected for a quadratic equation to represent an ellipse.

3 Early No-Intersection Test

The axis-aligned bounding boxes for two ellipses can be computed and tested for overlap. If they do not overlap, the ellipses cannot intersect. The bounding boxes are computed by locating extreme ellipse points in the coordinate axis directions. If multiple queries are used for a collection of ellipses, the axis-aligned bounding boxes can be precomputed for the ellipses and stored with the ellipse structure.

3.1 Axis-Aligned Bounding Box from Standard Form

Generally, let $\boldsymbol{N}$ be a unit-length vector. The ellipse has two extreme points on the line with origin $\boldsymbol{C}$ and direction $\boldsymbol{N}$. The quadratic function $Q(\boldsymbol{X}) = (\boldsymbol{X} - \boldsymbol{C})^{\mathsf{T}}M(\boldsymbol{X} - \boldsymbol{C}) - 1$ implicitly defines an ellipse as the level curve $Q(\boldsymbol{X}) = 0$. The gradient of $Q(\boldsymbol{X})$ is an outer-pointing normal vector at the ellipse point $\boldsymbol{X}$.

The gradient is $\nabla Q(\boldsymbol{X}) = 2M(\boldsymbol{X} - \boldsymbol{C})$. We wish to find ellipse points for which $\nabla Q(\boldsymbol{X})$ is parallel to $\boldsymbol{N}$. Specifically, we want $M(\boldsymbol{X} - \boldsymbol{C}) = t\boldsymbol{N}$ for two nonzero scalars t. Solve for $\boldsymbol{X} - \boldsymbol{C} = tM^{-1}\boldsymbol{N}$ and substitute this into the quadratic function to obtain $t^2\boldsymbol{N}^{\mathsf{T}}M^{-1}\boldsymbol{N} = 1$. We can solve for t to obtain the extreme points

$$\boldsymbol{X} = \boldsymbol{C} \pm M^{-1}\boldsymbol{N}/\sqrt{\boldsymbol{N}^{\mathsf{T}}M^{-1}\boldsymbol{N}} \tag{6}$$

The distance in the $\boldsymbol{N}$ direction from the center $\boldsymbol{C}$ to an extreme point $\boldsymbol{X}$ is

$$\left|\boldsymbol{N}^{\mathsf{T}}(\boldsymbol{X} - \boldsymbol{C})\right| = \boldsymbol{N}^{\mathsf{T}}M^{-1}\boldsymbol{N}/\sqrt{\boldsymbol{N}^{\mathsf{T}}M^{-1}\boldsymbol{N}} = \sqrt{\boldsymbol{N}^{\mathsf{T}}M^{-1}\boldsymbol{N}} \tag{7}$$

The extreme values for an axis-aligned bounding box are provided by the directions $\boldsymbol{N} = (1, 0)$ and $\boldsymbol{N} = (0, 1)$. Let $\widehat{\boldsymbol{U}} = (u_0, u_1)$ and $\widehat{\boldsymbol{V}} = (-u_1, u_0)$. The extreme offsets are

$$\pm\sqrt{\frac{m_{11}}{\det(M)}}\ (1, 0),\ \ \pm\sqrt{\frac{m_{00}}{\det(M)}}\ (0, 1) \tag{8}$$

where the formula for M^{-1} from equation (3) has been used.

Listing 1 contains C++ code for the no-intersection test.

Listing 1. C++ code for applying an early no-intersection test for the intersection of two ellipses.

```
template <typename T>
struct Ellipse2<T>
{
    // The comments contain the symbols used in equation (1).
    Vector2<T> center; // C
    std::array<Vector2<T>, 2> axis; // Û and V̂
    std::array<T, 2> extent; // a and b

    void GetStandardForm(Vector2<T>& C, Matrix2x2<T>& M) const
    {
        Matrix2x2<T> UUTrn = OuterProduct(axis[0], axis[0]);
        Matrix2x2<T> VVTrn = OuterProduct(axis[1], axis[1]);
        T USqrLen = Trace(UUTrn);
        T aSqr = extent[0] * extent[0];
        T bSqr = extent[1] * extent[1];
        C = center;
        M = (UUTrn / aSqr + VVTrn / bSqr) / USqrLength;
    }
};

template <typename T>
struct AlignedBox2<T>
{
    std::array<T, 2> min, max;

    bool Intersects(AlignedBox2<T> const& other) const
    {
        for (size_t i = 0; i < 2; ++i)
        {
            if (max[i] < other.min[i] || min[i] > other.max[i])
            {
                return false;
            }
        }
        return true;
    }
};
```

```
template <typename T>
void ComputeAlignedBox(Vector2<T> const& C, Matrix2x2<T> const& M, AlignedBox2<T>& box)
{
    T detM = M(0,0) * M(1,1) - M(0,1) * M(0,1);
    T xDistance = std::sqrt(M(1,1) / detM);
    T yDistance = std::sqrt(M(0,0) / detM);
    box.min[0] = C[0] - xDistance;
    box.max[0] = C[0] + xDistance;
    box.min[1] = C[1] - yDistance;
    box.max[1] = C[1] + yDistance;
}

template <typename T>
void ComputeAlignedBox(Ellipse2<T> const& ellipse, AlignedBox2<T>& box)
{
    Vector2<T> C{};
    Matrix2x2<T> M{};
    ellipse.GetStandardForm(C, M);
    ComputeAlignedBox(C, M, box);
}

template <typename T>
bool NoIntersection(Vector2<T> const& C0, Matrix2x2<T> const& M0, Vector2<T> const& C1, Matrix2x2<T> const& M1)
{
    AlignedBox2<T> box0{}, box1{};
    ComputeAlignedBox(C0, M0, box0);
    ComputeAlignedBox(C1, M1, box1);
    return !box0.Intersects(box1);
}

template <typename T>
bool NoIntersection(Ellipse2<T> const& ellipse0, Ellipse2<T> const& ellipse1)
{
    AlignedBox2<T> box0{}, box1{};
    ComputeAlignedBox(ellipse0, box0);
    ComputeAlignedBox(ellipse1, box1);
    return !box0.Intersects(box1);
}
```

3.2 Axis-Aligned Bounding Box from Quadratic Equation

The axis-aligned bounding box also can be computed from the quadratic equation (4). The term $(q_2+q_4x_0)x_1$ can be eliminated from the equation by an affine transformation $(x_0, x_1) = (y_0, y_1 - (q_2 + q_4y_0)/(2q_5)$ which leads to

$$\begin{aligned} \widetilde{Q}(y_0, y_1) &= 4q_5Q(x_0, x_1) \\ &= \left[(4q_0q_5 - q_2^2) + (4q_1q_5 - 2q_2q_4)y_0 + (4q_3q_5 - q_4^2)y_0^2\right] + 4q_5^2y_1^2 \\ &= \phi(y_0) + 4q_5^2y_1^2 \end{aligned} \tag{9}$$

where the last equality defines the polynomial $\phi(y_0)$. The degree of $\phi(y_0)$ is 2 because $q_3 > 0$, $q_5 > 0$, and $4q_3q_5 - q_4^2 > 0$. In order for $\widetilde{Q}(y_0, y_1) = 0$ to represent an ellipse, there must be points (y_0, y_1) that solve the equation which means y_0-values of the ellipse points must satisfy $\phi(y_0) \le 0$. Moreover, $\phi(y_0)$ must have 2 distinct real-valued roots. Applying the formulas in equation (5) and using some algebraic manipulation,

$$\phi(y_0) = 4\left(\det(M)(y_0 - c_0)^2 - m_{11}\right) \tag{10}$$

Computing the roots of $\phi(y_0)$, the y_0-extremes for the axis-aligned bounding box are $c_0 \pm \sqrt{m_{11}/\det(M)}$, which agrees with the first equation in (8).

Similarly, the affine transformation $(x_0, x_1) = (y_0 - (q_1 + q_4 y_1)/(2q_3), y_1)$ leads to

$$\begin{aligned}\overline{Q}(y_0, y_1) &= 4q_3 Q(x_0, x_1) \\ &= 4q_3^2 y_0^2 + \left[(4q_0 q_3 - q_1^2) + (4q_2 q_3 - 2q_1 q_4) y_1 + (4q_3 q_5 - q_4^2) y_1^2\right] \\ &= 4q_3^2 y_0^2 + \psi(y_1)\end{aligned} \tag{11}$$

where the last equality defines the polynomial $\psi(y_1)$. The degree of $\psi(y_1)$ is 2 because $q_3 > 0$, $q_5 > 0$, and $4q_3 q_5 - q_4^2 > 0$. In order for $\overline{Q}(y_0, y_1) = 0$ to represent an ellipse, there must be points (y_0, y_1) that solve the equation which means y_1-values of the ellipse points must satisfy $\psi(y_1) \le 0$. Moreover, $\psi(y_1)$ must have 2 distinct real-valued roots. Applying the formulas in equation (5) and using some algebraic manipulation,

$$\psi(y_1) = 4\left(\det(M)(y_1 - c_1)^2 - m_{00}\right) \tag{12}$$

Computing the roots of $\psi(y_1)$, the y_1-extremes for the axis-aligned bounding box are $c_1 \pm \sqrt{m_{00}/\det(M)}$, which agrees with the second equation in (8).

4 Solving Two Quadratic Equations in Two Unknowns

For $i \in \{0, 1\}$ let the ellipses have standard forms $(\boldsymbol{X} - \boldsymbol{C}_i)^{\mathsf{T}} M_i (\boldsymbol{X} - \boldsymbol{C}_i) = 1$. If $\boldsymbol{C}_0 = \boldsymbol{C}_1$ and $M_0 = M_1$, the ellipses are identical. An implemention will report this configuration. For the remainder of this document, the assumption is that the ellipses are not identical.

Each of the standard forms can be converted to quadratic form using equation (4),

$$\begin{aligned}F(x_0, x_1) &= f_0 + f_1 x_0 + f_2 x_1 + f_3 x_0^2 + f_4 x_0 x_1 + f_5 x_1^2 = 0 \\ G(x_0, x_1) &= g_0 + g_1 x_0 + g_2 x_1 + g_3 x_0^2 + g_4 x_0 x_1 + g_5 x_1^2 = 0\end{aligned} \tag{13}$$

which implicitly define the ellipses. The coefficients satisfy $f_3 > 0$, $f_5 > 0$, $4f_3 f_5 - f_4^2 > 0$, $g_3 > 0$, $g_5 > 0$, and $4g_3 g_5 - g_4^2 > 0$. The x_1^2 term can be eliminated by

$$\begin{aligned}0 &= E(x_0, x_1) \\ &= g_5 F(x_0, x_1) - f_5 G(x_0, x_1) \\ &= (f_0 g_5 - f_5 g_0) + (f_1 g_5 - f_5 g_1) x_0 + (f_2 g_5 - f_5 g_2) x_1 + (f_3 g_5 - f_5 g_3) x_0^2 + (f_4 g_5 - f_5 g_4) x_0 x_1 \\ &= e_0 + e_1 x_0 + e_2 x_1 + e_3 x_0^2 + e_4 x_0 x_1 \\ &= (e_0 + e_1 x_0 + e_3 x_0^2) + (e_2 + e_4 x_0) x_1 \\ &= \varepsilon_0(x_0) + \varepsilon_1(x_0) x_1\end{aligned} \tag{14}$$

where the fourth equality defines the e_i for $0 \le i \le 4$ and the last equality defines the polynomials $\varepsilon_j(x_0)$ for $0 \le j \le 1$.

The construction of intersection points requires analyzing the polynomials $\varepsilon_i(x_0)$. The cases to consider are $e_4 = 0$ and $e_2 = 0$, $e_4 = 0$ and $e_2 \ne 0$, and $e_4 \ne 0$. It is possible to present algebraic constructions of intersection points for these cases, but Section 3.2 provides a hint that simplifies the constructions. In that section an affine transformation is applied that consists of a translation and a shear.

5 Affine Transformation to Simplify the Analysis

The first ellipse in standard form is $(\boldsymbol{X}-\boldsymbol{C}_0)^{\mathsf{T}}M_0(\boldsymbol{X}-\boldsymbol{C}_0)=1$. Let the entries of the matrix be $M_0=[m_{0,ij}]$. The LDL^{T} decomposition can be used to factor $M_0=LDL^{\mathsf{T}}$ where

$$L=\begin{bmatrix}1&0\\ \ell&1\end{bmatrix}=\begin{bmatrix}1&0\\ m_{0,01}/m_{0,00}&1\end{bmatrix},\ D=\begin{bmatrix}d_0&0\\0&d_1\end{bmatrix}=\begin{bmatrix}m_{0,00}&0\\0&(m_{0,00}m_{0,11}-m_{0,01}^2)/m_{0,00}\end{bmatrix} \tag{15}$$

The ellipse equation becomes

$$\left(L^{\mathsf{T}}(\boldsymbol{X}-\boldsymbol{C}_0)\right)^{\mathsf{T}}D\left(L^{\mathsf{T}}(\boldsymbol{X}-\boldsymbol{C}_0)\right)=1 \tag{16}$$

Define the change of variables $\boldsymbol{Y}=L^{\mathsf{T}}(\boldsymbol{X}-\boldsymbol{C}_0)$. This is an affine transformation consisting of a translation and a shear. The ellipse equation becomes

$$\boldsymbol{Y}^{\mathsf{T}}D\boldsymbol{Y}=1 \tag{17}$$

which represents an axis-aligned ellipse centered at the origin of the $\boldsymbol{Y}$-coordinate system. The second ellipse in standard form is $(\boldsymbol{X}-\boldsymbol{C}_1)^{\mathsf{T}}M_1(\boldsymbol{X}-\boldsymbol{C}_1)=1$. Substituting $\boldsymbol{X}=L^{-\mathsf{T}}\boldsymbol{Y}+\boldsymbol{C}_0$ into this equation,

$$\left(L^{-\mathsf{T}}\boldsymbol{Y}-\boldsymbol{K}\right)^{\mathsf{T}}M_1\left(L^{-\mathsf{T}}\boldsymbol{Y}-\boldsymbol{K}\right)=1 \tag{18}$$

where L^{-T} is the transpose of the inverse L^{-1} and $\boldsymbol{K}=\boldsymbol{C}_1-\boldsymbol{C}_0=(k_0,k_1)$.

Converting to quadratic equations, we have

$$\begin{aligned}F(y_0,y_1)&=-1+d_0y_0^2+d_1y_1^2=0\\ G(y_0,y_1)&=g_0+g_1y_0+g_2y_1+g_3y_0^2+g_4y_0y_1+g_5y_1^2=0\end{aligned} \tag{19}$$

where

$$\begin{aligned}g_0&=-1+k_0(k_0m_{1,00}+k_1m_{1,01})+k_1(k_0m_{1,01}+k_1m_{1,11})\\ g_1&=-2(k_0m_{1,00}+k_1m_{1,01})\\ g_2&=2[(k_0m_{1,00}+k_1m_{1,01})\ell-(k_0m_{1,01}+k_1m_{1,11})]\\ g_3&=m_{1,00}\\ g_4&=-2(m_{1,00}\ell-m_{1,01})\\ g_5&=m_{1,00}\ell^2-2m_{1,01}\ell+m_{1,11}\end{aligned} \tag{20}$$

Compare these quadratic equations to those of (13). In the $\boldsymbol{Y}$-coordinate system, equation (14) becomes the following. It does not matter whether E or $-E$ is used, so to eliminate a lot of negation operations this equation uses the negative of that in (13),

$$\begin{aligned}0&=E(y_0,y_1)\\ &=d_1G(y_0,y_1)-g_5F(y_0,y_1)\\ &=(g_5+d_1g_0)+(d_1g_1)y_0+(d_1g_2)y_1+(d_1g_3-d_0g_5)y_0^2+(d_1g_4)y_0y_1\\ &=e_0+e_1y_0+e_2y_1+e_3y_0^2+e_4y_0y_1\\ &=(e_0+e_1y_0+e_3y_0^2)+(e_2+e_4y_0)y_1\\ &=\varepsilon_0(y_0)+\varepsilon_1(y_0)y_1\end{aligned} \tag{21}$$

where $e_0 = g_5 + d_1 g_0$, $e_1 = d_1 g_1$, $e_2 = d_1 g_2$, $e_3 = d_1 g_3 - d_0 g_5$, and $e_4 = d_1 g_4$.

Symbolically, we can solve for

$$y_1 = -\varepsilon_0(y_0)/\varepsilon_1(y_0) = -\frac{e_0 + e_1 y_0^2 + e_3 y_0^2}{e_2 + e_4 y_0} \tag{22}$$

substitute this into the equation $F(y_0, y_1) = 0$ and eliminate the denominator of the fraction to obtain a polynomial of degree at most 4,

$$H(y_0) = (e_2 + e_4 y_0)^2(-1 + d_0 y_0^2) + d_1(e_0 + e_1 y_0^2 + e_3 y_0^2)^2 \tag{23}$$

whose roots r_0 can be used to generate points (r_0, r_1), where $r_1 = -\varepsilon_0(r_0)/\varepsilon_1(r_0)$. An analysis of the e_i is required to determine whether or not the divisor in the y_1-equation is nonzero. Even if it is nonzero, the division can be a major source of rounding errors when using floating-point arithmetic.

6 Case e_4 is Zero and e_2 is Zero

Let $e_4 = 0$ and $e_2 = 0$, which imply $g_4 = 0$ and $g_2 = 0$. The G-ellipse is axis aligned. The division in equation (22) cannot be performed because the denominator is identically zero. The polynomial of equation (23) becomes

$$H(y_0) = e_0 + e_1 y_0 + e_3 y_0^2 \tag{24}$$

Further analysis is required for e_3, e_2, and e_0.

The condition $g_4 = 0$ implies

$$m_{1,00}\ell - m_{1,01} = 0 \tag{25}$$

Using the definition $\ell = m_{0,01}/m_{0,00}$ and knowing that $m_{0,00} > 0$, we obtain $m_{1,00}m_{0,01} - m_{1,01}m_{0,00} = 0$. In turn this implies $(m_{1,00}, m_{1,01})$ and $(m_{0,00}, m_{0,01})$ are parallel, say,

$$(m_{1,00}, m_{1,01}) = s(m_{0,00}, m_{0,01}) \tag{26}$$

for some $s > 0$. The positivity of s is implied by M_0 and M_1 being positive definite matrices, in which case $m_{0,00} > 0$ and $m_{1,00} > 0$.

The condition $g_2 = 0$ implies

$$\begin{aligned} 0 &= (k_0 m_{1,00} + k_1 m_{1,01})\ell - (k_0 m_{1,01} + k_1 m_{1,11}) \\ &= k_0(m_{1,00}\ell - m_{1,01}) + k_1(m_{1,01}\ell - m_{1,11}) \qquad \text{[from equation (25)]} \\ &= k_1(m_{1,01}\ell - m_{1,11}) \end{aligned} \tag{27}$$

Either $k_1 = 0$ or $m_{1,01}\ell - m_{1,11} = 0$.

6.1 Case e_3 is Zero

Let $e_3 = 0$. The H-polynomial is linear

$$H(y_0) = e_0 + e_1 y_0 \tag{28}$$

and an implication is $d_0 g_3 - d_1 g_5 = 0$. This implies that (g_3, g_5) and (d_0, d_1) are parallel, say $(g_3, g_5) = t(d_0, d_1)$. We know that $g_3 = m_{1,00}$, $d_0 = m_{0,00}$, and from equation (26) $m_{1,00} = s m_{0,00}$. Consequently, $t = s$ and $g_5 = s d_1$. Using the definitions of g_5 and d_1,

$$\begin{aligned} s(m_{0,00}m_{0,11} - m_{0,01}^2)/m_{0,00} &= m_{1,00}\ell^2 - 2m_{1,01}\ell + m_{1,11} \\ &= (m_{1,00}\ell - m_{1,01})\ell - m_{1,01}\ell + m_{1,11} \\ &= -m_{1,01}\ell + m_{1,11} \qquad \text{[from equation (25)]} \\ &= -s m_{0,01}(m_{0,01}/m_{0,00}) + m_{1,11} \qquad \text{[from equation (26)]} \end{aligned} \tag{29}$$

Solving for $m_{1,11}$, we obtain

$$m_{1,11} = s m_{0,01}(m_{0,01}/m_{0,00}) + s(m_{0,00}m_{0,11} - m_{0,01}^2)/m_{0,00} = s m_{0,11} \tag{30}$$

Combined with previous derivations, we now know that $M_1 = sM_0$. Moreover, equation (27) becomes

$$\begin{aligned} 0 &= k_1(m_{1,01}\ell - m_{1,11}) \\ &= k_1 s(m_{0,01}\ell - m_{0,11}) \qquad \text{[from equations (26) and (30)]} \\ &= k_1 s(m_{0,01}^2 - m_{0,00}m_{11})/m_{00} \\ &= -k_1 s \det(M_0) \end{aligned} \tag{31}$$

We know $s > 0$ and $\det(M_0) > 0$, so it must be that $k_1 = 0$. The coefficients g_0 and g_1 reduce to

$$g_0 = -1 + k_0^2 m_{1,00} = -1 + s d_0 k_0^2, \quad g_1 = -2k_0 m_{1,00} = -2s d_0 k_0 \tag{32}$$

6.1.1 Case e_1 is Zero

If $e_1 = 0$, then $g_1 = 0$ which implies $k_0 = 0$ and the G-polynomial is $G(y_0, y_1) = -1 + s(d_0 y_0^2 + d_1 y_1^2)$. The F-polynomial is $F(y_0, y_1) = -1 + d_0 y_0^2 + d_1 y_1^2$. Substituting $F = 0$ into $G = 0$, we obtain $s - 1 = 0$. If $s = 1$, the ellipses are identical, a condition that should be tested for first in an implementation. If $s \neq 1$, the ellipses do not intersect. They have the same center and same shape, but the extent vectors are parallel and not equal.

6.1.2 Case e_1 is not Zero

If $e_1 \neq 0$, then $g_1 \neq 0$ which implies $k_0 \neq 0$ and the G-polynomial is $G(y_0, y_1) = -1 + s(d_0(y_0 - k_0)^2 + d_1 y_1^2)$. The F-polynomial is $F(y_0, y_1) = -1 + d_0 y_0^2 + d_1 y_1^2$. Substituting $F = 0$ into $G = 0$ produces the equation $-1 + s(1 + d_0 k_0^2 - 2d_0 k_0 y_0) = 0$. The solution is

$$y_0 = \frac{d_0 k_0^2 + 1 - 1/s}{2 d_0 k_0} = \frac{k_0}{2} + \frac{1 - 1/s}{2 d_0 k_0} \tag{33}$$

which gives geometric information about the ellipses, something the solution $y_0 = -e_0/e_1$ does not immediately provide. The denominator is not zero, but when it is nearly zero and floating-point arithmetic is used for computing, rounding errors can lead to an inaccurate y_0 value.

When $s = 1$, the G-ellipse is a translation of the F-ellipse by an amount k_0 in the y_0-direction. There are 2 intersections occurring when $y_0 = k_0/2$, which is clear by the symmetry of the geometric configuration.

When $s > 1$, if the G-ellipse were to be centered at $(0, 0)$, it is strictly inside the F-ellipse. As the G-ellipse is translated by k_0 in the y_0-direction, eventually it intersects the F-ellipse at $(\text{Sign}(k_0)a_0, 0)$, where a_0 is the extent of the F-ellipse associated with the y_0-axis. Recall that $1/a_0^2 = d_0$. Increasing $|k_0|$ farther leads to 2 intersection points. Define the function $\phi(z) = z/2 + c/z$ where $c = (1 - 1/s)/(2d_0) > 0$. The function has a local minimum of $\sqrt{2c}$ occurring when $z = \sqrt{2c}$ and a local maximum of $-\sqrt{2c}$ when $z = -\sqrt{2c}$. These imply that at the intersection points, $\sqrt{(1 - 1/s)/d_0} \le |\phi(k_0)| = |y_0| \le a_0$

When $s < 1$, if the G-ellipse were to be centered at $(0, 0)$, it is strictly outside the F-ellipse. As the G-ellipse is translated by k_0 in the y_0-direction, eventually it intersects the F-ellipse at $(-\text{Sign}(k_0)a_0, 0)$. Shifting some more (by increasing $|k_0|$) leads to 2 intersection points. The function $\phi(z)$ has $c < 0$, so there are no local extrema. However, at the intersection points, $|y_0| \le a_0$.

In terms of the e-coefficients, $y_0 = -e_0/e_1$. Numerically we must be concerned about rounding errors when e_1 is nearly zero. Given the bounds $|-e_0/e_1| = |y_0| \le a_0$, a necessary condition for the existence of intersection points is $e_1^2 - d_0 e_0^2 \ge 0$. In an implementation, this condition can be tested first, and if true, the division is then computed.

6.2 Case e_3 is not Zero

The H-polynomial is quadratic,

$$H(y_0) = e_0 + e_1 y_0 + e_3 y_0^2 \tag{34}$$

For each root r_0, the y_1 values are computed by solving $F(r_0, y_1) = 0$. Define $\lambda = 1 - d_0 r_0^2$. If $\lambda < 0$, there are no points of intersection corresponding to r_0. If $\lambda = 0$, there is 1 point of intersection $y_1 = 0$. If $\lambda > 0$, there are 2 points of intersection (r_0, r_1) where $r_1 = \pm\sqrt{\lambda/d_1}$.

7 Case e_4 is Zero and e_2 is not Zero

Because $e_4 \ne 0$, both ellipses are axis aligned. We can solve equation (21) for $y_1 = -(e_0 + e_1 y_0 + e_3 y_0^2)/e_2$. Substituting this expression into $F(y_0, y_1) = 0$ and multiplying by e_2^2 leads to an H-polynomial that is quartic or quadratic depending on whether e_3 is not zero or is zero.

7.1 Case e_3 is Zero

The H-polynomial is quadratic,

$$H(y_0) = e_2^2(-1 + d_0 y_0^2) + d_1(e_0 + e_1 y_0)^2 \tag{35}$$

The coefficient of y_0^2 is $d_0 e_2^2 + d_1 e_1^2 > 0$. The positivity is guaranteed because $d_0 > 0$, $e_2^2 > 0$, and $d_1 e_1^2 \ge 0$.

Theoretically, for each root r_0 of $H(y_0)$, compute the corresponding $r_1 = -(e_0 + e_1 r_0)/e_2$ to obtain an intersection point (r_0, r_1).

Numerically when computing with floating-point arithmetic, if e_2 is nearly zero, the division by it can have rounding errors that lead to an inaccurate r_1. To avoid the division, for each root r_0 of $H(y_0)$, solve

$F(r_0, y_1) = 0$ for values r_1. Define $\lambda = 1 - d_0 r_0^2$. If $\lambda < 0$, there are no points of intersection corresponding to r_0. If $\lambda = 0$, there is 1 point of intersection $y_1 = 0$. If $\lambda > 0$, there is only 1 point of intersection (not 2) because we know theoretically $r_1 = -(e_0 + e_1 r_0)/e_2$. The question is which of the two roots $\pm\sqrt{\lambda/d_1}$ corresponds to r_1. The simplest numerical decision is to evaluate $t_0 = |(e_0 + e_1 r_0) - e_2\sqrt{\lambda/d_1}|$ and $t_1 = |(e_0 + e_1 r_0) + e_2\sqrt{\lambda/d_1}|$ and then choose r_1 to be that root which produces the minimum of t_0 and t_1.

7.2 Case e_3 is not Zero

The H-polynomial is quartic,

$$H(y_0) = e_2^2(-1 + d_0 y_0^2) + d_1(e_0 + e_1 y_0 + e_3 y_0^2)^2 \tag{36}$$

The coefficient of y_0^4 is $d_1 e_3^2 > 0$. The positivity is guaranteed because $d_1 > 0$ and $e_3^2 > 0$.

Theoretically, for each root r_0 of $H(y_0)$, compute the corresponding $r_1 = -(e_0 + e_1 r_0 + e_3 r_0^2)/e_2$ to obtain an intersection point (r_0, r_1).

Numerically when computing with floating-point arithmetic, if e_2 is nearly zero, the division by it can have rounding errors that lead to an inaccurate r_1. To avoid the division, for each root r_0 of $H(y_0)$, solve $F(r_0, y_1) = 0$ for values r_1. Define $\lambda = 1 - d_0 r_0^2$. If $\lambda < 0$, there are no points of intersection corresponding to r_0. If $\lambda = 0$, there is 1 point of intersection $y_1 = 0$. If $\lambda > 0$, there is only 1 point of intersection (not 2) because we know theoretically $r_1 = -(e_0 + e_1 r_0 + e_3 r_0^2)/e_2$. The question is which of the two roots $\pm\sqrt{\lambda/d_1}$ corresponds to r_1. The simplest numerical decision is to evaluate $t_0 = |(e_0 + e_1 r_0 + e_3 r_0^2) - e_2\sqrt{\lambda/d_1}|$ and $t_1 = |(e_0 + e_1 r_0 + e_3 r_0^2) + e_2\sqrt{\lambda/d_1}|$ and then choose r_1 to be that root which produces the minimum of t_0 and t_1.

8 Case e_4 is not Zero

The second ellipse is not axis aligned in this case. We can solve equation (21) for $y_1 = -(e_0 + e_1 y_0 + e_3 y_0^2)/(e_2 + e_4 y_0)$. Substituting this expression into $F(y_0, y_1) = 0$ and multiplying by $(e_2 + e_4 y_0)^2$ to obtain the quartic polynomial

$$H(y_0) = (e_2 + e_4 y_0)^2(-1 + d_0 y_0^2) + d_1(e_0 + e_1 y_0 + e_3 y_0^2)^2 = 0 \tag{37}$$

The coefficient of y_0^4 is $d_0 e_4^2 + d_1 e_3^2 > 0$. The positivity is guaranteed because $d_0 > 0$, $e_4^2 > 0$, and $d_1 e_3^2 \geq 0$.

Similar to the case $e_2 = e_4 = 0$, the divisor $\delta = e_2 + e_4 r_0$ for a root r_0 of $H(y_0)$ can be exactly zero, in which case the division is not valid. If *delta* is not zero, then the division is allowed but numerically it can be nearly zero in which case the division should be avoided.

For each root r_0 of $H(y_0)$, solve $F(r_0, y_1) = 0$ for values r_1. Define $\lambda = 1 - d_0 r_0^2$. If $\lambda < 0$, there are no points of intersection. If $\lambda = 0$, there is 1 point of intersection $y_1 = 0$. Now consider $\lambda > 0$. If $\delta = 0$, there are 2 points of intersection with $\pm\sqrt{\lambda/d_1}$. If $\delta \neq 0$, the question is which of the two roots $\pm\sqrt{\lambda/d_1}$ corresponds to r_1. The simplest numerical decision is to evaluate $t_0 = |(e_0 + e_1 r_0 + e_3 r_0^2) - (e_2 + e_4 r_0)\sqrt{\lambda/d_1}|$ and $t_1 = |(e_0 + e_1 r_0 + e_3 r_0^2) + (e_2 + e_4 r_0)\sqrt{\lambda/d_1}|$ and then choose r_1 to be that root which produces the minimum of t_0 and t_1.

Note that when $e_2 + e_4 r_0 = 0$, then also $e_0 + e_1 r_0 + e_3 r_0^2 = 0$. This can happen only when $e_2^2 e_3 - e_1 e_2 e_4 + e_0 e_4^2 = 0$. It is unknown whether this can happen theoretically, but the code handles this anyway.

9 Eliminating Divisions

The standard form for the ellipses is equation (1), which involves divisions. Using $i \in \{0,1\}$ to index the ellipses, the divisions can be eliminated by multiplying the ellipse equation by $r_i^2 = a_i^2 b_i^2 |\widehat{\boldsymbol{U}}_i|^2$ to obtain

$$(\boldsymbol{X} - \boldsymbol{C}_i)^{\mathsf{T}} \widehat{M}_i (\boldsymbol{X} - \boldsymbol{C}_i) = r_i^2 \tag{38}$$

where $\widehat{M}_i = r_i^2 M_i$.

The LDL^{T} matrices of equation (15) can be replaced by $\widehat{L} = m_{0,00} L$ and $\widehat{D} = m_{0,00} D$. Equation (16) becomes

$$\left(\widehat{L}^{\mathsf{T}}(\boldsymbol{X} - \boldsymbol{C}_0)\right)^{\mathsf{T}} \widehat{D} \left(\widehat{L}^{\mathsf{T}}(\boldsymbol{X} - \boldsymbol{C})\right) = m_{0,00}^3 r_0^2 \tag{39}$$

The affine transformation is now $\boldsymbol{Y} = \widehat{L}^{\mathsf{T}}(\boldsymbol{X} - \boldsymbol{C}_0)$ so that equation (17) becomes

$$\boldsymbol{Y}^{\mathsf{T}} \widehat{D} \boldsymbol{Y} = m_{0,00}^3 r_0^2 \tag{40}$$

and equation (18) becomes

$$\left(\widehat{L}^{-\mathsf{T}} \boldsymbol{Y} - \boldsymbol{\Delta}\right)^{\mathsf{T}} \widehat{M}_1 \left(\widehat{L}^{-\mathsf{T}} \boldsymbol{Y} - \boldsymbol{\Delta}\right) = r_1^2 \tag{41}$$

10 Implementation Issues

The previous discussion was theoretical in the sense that (error-free) real-valued arithmetic is used in the derivations. Computing with floating-point arithmetic, an implementation must be aware of the potential problems caused by rounding errors and subtractive cancellation. It is not possible to avoid rounding errors, even if one were to use rational arithmetic, because root finding for quadratic and quartic polynomials produces roots that are not rational. That said, the goal is to minimize the effects of rounding errors in order to avoid computing points of intersection that are not accurate enough to be acceptable.

10.1 Using the Unnormalized Ellipse Axes

We already saw that the standard form of an ellipse shown in equation (1) has rounding errors when the unit-length ellipse axis directions are computed via normalization. To avoid this problem, use non-unit-length ellipse axis directions. It is also possible to normalize the axis directions and pass them to the ellipse construction, but the intersection code will still apply the division by the squared length.

10.2 Computing Coefficients

The setup of Section 5 can be computed using rational arithmetic when considering the input ellipse parameters as error-free floating-point numbers. The parameters are necessarily rational numbers. The affine transformation is also computable using rational arithmetic, which means f_i and g_j can be computed without errors. And the coefficients of the polynomial $H(y_0)$ can be computed using rational arithmetic. The root finding, however, can only estimate non-rational roots. It is possible that H has some rational roots, and a

good root finder can compute them exactly. This happens when roots are repeated, which in the context of computing intersection points of ellipses occurs when the ellipses have tangential contact.

The computations can also be performed using floating-point arithmetic, albeit with rounding errors. Several steps in the process are sensitive to rounding errors, so the implementation must handle these carefully.

10.3 Robust Computation of Determinant-Like Expressions

The coefficients of $E(y_0, y_1)$ in equation (21) are expressions of the form u*v, u*v+w, and u*v+w*z for floating-point numbers u, v, w, and z. When computing with floating-point arithmetic, we need to worry about rounding errors associated with the expressions. The expression u*v is computed using a floating-point multiplication, where the result is the floating-point number closest to the theoretical value of the expression. If the expression u*v+w is computed with a floating-point multiplication and a floating-point division, each operation involves rounding. Current floating-point hardware provides a *fused multiply-add* instruction, call it fma(u,v,w), that involves only a single rounding step. The result is the floating-point number closest to the theoretical value of the expression, which is better than having two rounding steps.

Robust floating-point computation of the expression uv+wz is more complicated. The näive approach involves two floating-point multiplications and one floating-point addition for a total of three rounding steps. Worse is that if uv and wz have opposite signs but nearly the same magnitude, subtractive cancellation of significant bits can occur which leads to inaccurate results. The problem is resolved by using fused multiply-add instructions. See the document On the Cost of Floating-Point Computation Without Extra-Precise Arithmetic by Turing Award winner Professor William Kahan. Matt Pharr has a concise summary of a portion of Professor Kahan's document regarding rounding errors for differences of products; see Accurate Differences of Products with Kahan's Algorithm. The implementation is shown in Listing 2, together with a robust sum of products,

Listing 2. Robust computation of a difference of products where T is a floating-point type.

```
// Compute robustly the difference of products uv − wz.
template <typename T>
T DifferenceOfProducts(T u, T v, T w, T z)
{
    T productWZ = w * z;
    T roundingError = std::fma(−w, z, productWZ);
    return std::fma(u, v, −productWZ) + roundingError;
}

// Compute robustly the sum of products uv + wz.
template <typename T>
T SumOfProducts(T u, T v, T w, T z)
{
    T productWZ = w * z;
    T roundingError = std::fma(w, z, −productWZ);
    return std::fma(u, v, productWZ) + roundingError;
}
```

10.4 Implementation

The implementation of the find-intersection query is found in IntrEllipse2Ellipse2.h. Version 6.0.2022.11.11 contained an implementation that is described in the document Intersection of Ellipses as well as in the book *Robust and Error-Free Geometric Computing* [1, Section 8.6].

The intent of the PDF and book are to describe how to compute robustly the intersections when using a mixture of floating-point arithmetic and rational arithmetic. However, the code was not robust with only floating-point arithmetic. The main source of rounding errors in some cases occurs because of the divisions $w = -e(x)/d(x)$ in equation (14) of the PDF or equation (8.40) of the book. These errors sometimes led to significantly inaccurate points of intersection.

Version 6.0.2023.06.14 of the header file has a robust implementation for floating-point arithmetic. The divisions were avoided, as described in the current PDF (this document). The unit tests and end-to-end tests appear to produce accurate points of intersection. An interactive sample application is found in the folder GeometricTools/GTE/Samples/Mathematics/IntersectEllipses.

References

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