Distance Between Point and Triangle in 3D

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

Created: September 28, 1999
Last Modified: September 11, 2020

Contents

1 Mathematical Formulation

The problem is to compute the minimum distance between a point $\mathbf{P}$ and a triangle $\mathbf{T}(s,t) = \mathbf{B} + s\mathbf{E}_0 + t\mathbf{E}_1$ for $(s,t) \in D = \{(s,t) : s \geq 0, t \geq 0, s+t \leq 1\}$. The minimum distance is computed by locating the values $(\tilde{s}, \tilde{t}) \in D$ so that $\mathbf{T}(\tilde{s}, \tilde{t})$ is the triangle point closest to $\mathbf{P}$.

The squared-distance from $\mathbf{P}$ to any point $\mathbf{T}(s,t)$ on the triangle is the quadratic function

$$Q(s,t) = |\mathbf{T}(s,t) - \mathbf{P}|^2 = as^2 + 2bst + ct^2 + 2ds + 2et + f \tag{1}$$

for $(s,t) \in D$ and where $a = \mathbf{E}_0 \cdot \mathbf{E}_0$, $b = \mathbf{E}_0 \cdot \mathbf{E}_1$, $c = \mathbf{E}_1 \cdot \mathbf{E}_1$, $d = \mathbf{E}_0 \cdot (\mathbf{B} - \mathbf{P})$, $e = \mathbf{E}_1 \cdot (\mathbf{B} - \mathbf{P})$ and $f = (\mathbf{B} - \mathbf{P}) \cdot (\mathbf{B} - \mathbf{P})$. Quadratics are classified by the sign of $ac - b^2$, which for $Q(s,t)$ is

$$ac - b^2 = (\mathbf{E}_0 \cdot \mathbf{E}_0)(\mathbf{E}_1 \cdot \mathbf{E}_1) - (\mathbf{E}_0 \cdot \mathbf{E}_1)^2 = |\mathbf{E}_0 \times \mathbf{E}_1|^2 > 0 \tag{2}$$

The positivity is based on the assumption that the two edges $\mathbf{E}_0$ and $\mathbf{E}_1$ of the triangle are linearly independent, so their cross product is a nonzero vector. Also observe that

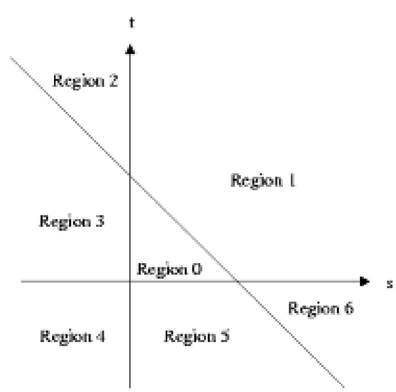
$$a - 2b + c = \mathbf{E}_0 \cdot \mathbf{E}_0 - 2\mathbf{E}_0 \cdot \mathbf{E}_1 + \mathbf{E}_1 \cdot \mathbf{E}_1 = |\mathbf{E}_0 - \mathbf{E}_1|^2 > 0 \tag{3}$$

The goal is to minimize $Q(s,t)$ over D. Because Q is a continuously differentiable function, the minimum occurs either at an interior point of D where the gradient $\nabla Q = 2(as + bt + d, bs + ct + e) = (0,0)$ or at a point on the boundary of D.

2 The Algorithm

The gradient of Q is zero only when $\bar{s} = (be - cd)/(ac - b^2)$ and $\bar{t} = (bd - ae)/(ac - b^2)$. If $(\bar{s}, \bar{t}) \in D$, then we have found the minimum of Q; otherwise, the minimum must occur on the boundary of the triangle. To find the correct boundary, consider the partitioning of the plane shown in Figure 1.

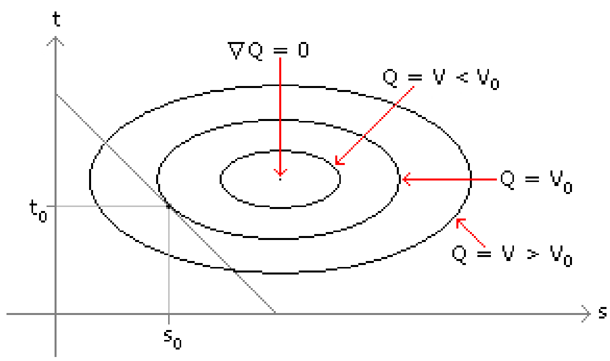
Figure 1. Partitioning of the st-plane by triangle domain D.



Region 0 is the triangular domain D of $Q(s,t)$. If $(\bar{s},\bar{t})$ is in region 0, the point $\mathbf{T}(\bar{s},\bar{t})$ closest to $\mathbf{P}$ is interior to the triangle.

Suppose $(\bar{s},\bar{t})$ is in region 1. The level curves of Q are those curves in the st-plane for which Q is a constant. The level curves are ellipses because the graph of Q is a paraboloid. At the point where $\nabla Q = (0,0)$, the level curve degenerates to a single point $(\bar{s},\bar{t})$. The global minimum of Q occurs there, call it $V_{\min}$. As the level values V increase from $V_{\min}$, the corresponding ellipses are increasingly farther away from $(\bar{s},\bar{t})$. There is a smallest level value V_0 for which the corresponding ellipse (implicitly defined by $Q = V_0$) just touches the triangle domain edge $s+t=1$ at a pair (s_0,t_0) where $s_0 \in [0,1]$ and $t_0 = 1-s_0$. For level values $V < V_0$, the corresponding ellipses do not intersect D. For level values $V > V_0$, portions of D lie inside the corresponding ellipses. In particular, any points of intersection of such an ellipse with the edge must have a level value $V > V_0$. Therefore, $Q(s,1-s) > Q(s_0,t_0)$ for $s \in [0,1]$ and $s \neq s_0$. The point (s_0,t_0) provides the minimum squared distance between $\mathbf{P}$ and the triangle. The triangle point is an edge point. Figure 2 illustrates the idea by showing various level curves.

Figure 2. Various level curves $Q(s,t) = V$.



An alternate way of visualizing where the minimum distance point occurs on the boundary is to intersect the graph of Q with the plane $s+t=1$. The curve of intersection is a parabola and is the graph of $F(s) = Q(s,1-s)$ for $s \in [0,1]$. Now the problem has been reduced by one dimension to minimizing a function $F(s)$ for $s \in [0,1]$. The minimum of $F(s)$ occurs either at an interior point of $[0,1]$, in which case $F'(s) = 0$ at that point, or at an endpoint $s=0$ or $s=1$. Figure 2 shows the case when the minimum occurs at an interior point. At that point the ellipse is tangent to the line $s+t=1$. In the endpoint cases, the ellipse may just touch one of the vertices of D, but not necessarily tangentially.

To distinguish between the interior-point and endpoint cases, the same partitioning idea applies in the 1-dimensional case. The interval $[0,1]$ partitions the real line into three intervals, $s<0$, $s \in [0,1]$ and $s>1$. Let $F'(\hat{s}) = 0$. If $\hat{s} < 0$, then $F(s)$ is an increasing function for $s \in [0,1]$. The minimum restricted to $[0,1]$ must occur at $s=0$, in which case Q attains its minimum at $(s,t) = (0,1)$. If $\hat{s} > 1$, then $F(s)$ is a decreasing function for $s \in [0,1]$. The minimum for F occurs at $s=1$ and the minimum for Q occurs at $(s,t) = (1,0)$. Otherwise, $\hat{s} \in [0,1]$, F attains its minimum at $\hat{s}$ and Q attains its minimum at $(s,t) = (\hat{s}, 1-\hat{s})$.

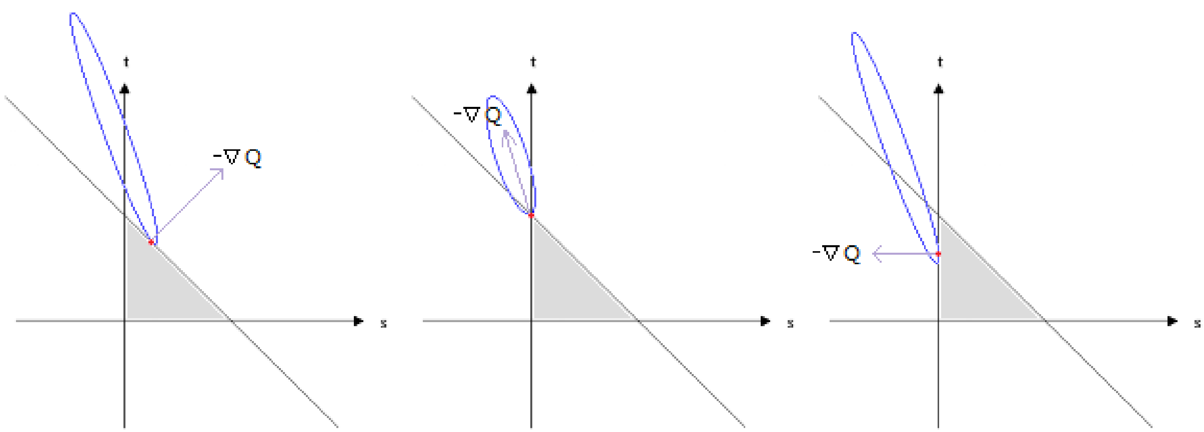
The occurrence of $(\bar{s},\bar{t})$ in region 3 or 5 is handled in the same way as when the global minimum is in region 1. If $(\bar{s},\bar{t})$ is in region 3, then the minimum occurs at $(0,t_0)$ for some $t_0 \in [0,1]$. If $(\bar{s},\bar{t})$ is in region 5, then the minimum occurs at $(s_0,0)$ for some $s_0 \in [0,1]$. Determining whether the first contact point is at an interior or endpoint of the appropriate interval is handled the same as discussed previously.

If $(\bar{s}, \bar{t})$ is in region 2, it is possible the level curve of Q that provides first contact with the triangle touches either edge $s + t = 1$ or edge $s = 0$. Because the global minimum occurs in region 2, the negative of the gradient at the corner $(0, 1)$ cannot point inside D. The geometric intuition for this is the following. There is a level curve of Q that just touches the triangle and for which the region inside the triangle and the region inside the level curve do not overlap. The level curve does one of the following.

1. Touches the edge for which $s + t = 1$.
2. Touches the corner at $(0, 1)$.
3. Touches the edge for which $s = 0$.

Figure 3 illustrates the three cases.

Figure 3. Three ways a level curve with center in region 2 touches the triangle. The left diagram shows the level curve touching the edge $s = 0$ tangentially. The middle diagram shows the level curve touching the edge at $(0, 1)$ nontangentially. The right diagram shows the level curve touching the edge $s + t = 1$ tangentially.



At any point on the level curve, $-\nabla Q$ is in the direction towards the inside of the ellipse, which implies it cannot point inside the triangle. To decide which of the these three cases occurs, we can analyze the signs of the directional derivates of Q along the two edges of the triangle sharing vertex $(0, 1)$.

1. For edge $s + t = 1$, an edge direction pointing away from the vertex is $(1, -1)/\sqrt{2}$. The directional derivative at the vertex in this direction is $((1, -1)/\sqrt{2}) \cdot \nabla Q(0, 1)$. If it is negative, the minimum must occur on the edge $s+t = 1$ at some $\tilde{s} > 0$. If $\tilde{s} < 1$, we have the situation illustrated in the left diagram of Figure 3. If $\tilde{s} \geq 1$, the minimum occurs at the vertex $(1, 0)$ where $\tilde{s} = 1$.
2. For edge $s = 0$, an edge direction pointing away from the vertex is $(0, -1)$. The directional derivative at the vertex in this direction is $(0, -1) \cdot \nabla Q(0, 1)$. If it is negative, the minimum must occur on the edge $s = 0$ at some $\tilde{t}$. If $0 < \tilde{t} < 1$, we have the situation illustrated in the right diagram of Figure 3. If $\tilde{t} > 1$, the minimum occurs at the vertex $(0, 1)$, which is the situation illustrated in the middle diagram of Figure 3. If $\tilde{t} \leq 0$, the minimum occurs at the vertex $(0, 0)$.

3. It is not geometrically possible for the two aforementioned directional derivatives to be negative at the same time. This allows an implementation to use a simple if-then-else statement applied to the sign of one of the directional derivatives.

It is sufficient to analyze the sign of $(1,-1)\cdot\nabla Q(0,1)$ because the normalization of the edge direction does not change the sign.

The same type of argument applies in region 6 at the vertex $(1,0)$. The edge direction for $s+t=1$ is $(-1,1)/\sqrt{2}$ and the directional derivative is $((-1,1)/\sqrt{2})\cdot\nabla Q(1,0)$. The edge direction for $t=0$ is $(-1,0)$ and the directional derivative if $(-1,0)\cdot\nabla Q(1,0)$. And the same type of argument applies in region 4 at the vertex $(0,0)$. The edge direction for $t=0$ is $(1,0)$ and the directional derivative is $(1,0)\cdot\nabla Q(0,0)$. The edge direction for $s=0$ is $(0,1)$ and the directional derivative is $(0,1)\cdot\nabla Q(0,0)$.

3 Implementation

The implementation of the algorithm is designed so that at most one floating point division is used when computing the minimum distance and corresponding closest points. Moreover, the division is deferred until it is needed. In some cases no division is needed. Quantities that are used throughout the code are computed first. In particular, the values computed are $\mathbf{D}=\mathbf{B}-\mathbf{P}$, $a=\mathbf{E}_0\cdot\mathbf{E}_0$, $b=\mathbf{E}_0\cdot\mathbf{E}_1$, $c=\mathbf{E}_1\cdot\mathbf{E}_1$, $d=\mathbf{E}_0\cdot\mathbf{D}$, $e=\mathbf{E}_1\cdot\mathbf{D}$ and $f=\mathbf{D}\cdot\mathbf{D}$. The code actually computes $\delta=|ac-b^2|$ rather than $\delta=ac-b^2$ because it is possible for small edge lengths that some floating point round-off errors lead to a small negative quantity for $ac-b^2$.

In the theoretical development, we computed $\bar{s}=(be-cd)/\delta$ and $(bd-ae)/\delta$ so that $\nabla Q(\bar{s},\bar{t})=(0,0)$. The location of the global minimum is then tested to see whether it is in the triangle domain D. If so, then we have already determined what we need to compute the minimum distance. If not, then the boundary of D must be tested. To defer the division by δ, the code instead computes $\bar{s}=be-cd$ and $\bar{t}=bd-ae$ and tests for containment in a scaled domain, $s\in[0,\delta]$, $t\in[0,\delta]$ and $s+t\leq\delta$. If in this set, the divisions are performed. If not, the boundary of the triangle is tested. The general outline of the conditionals for determining which region contains $(\bar{s},\bar{t})$ is shown in Listing 1.

Listing 1. The general outline for determining which region contains $(\bar{s}, \bar{t})$.

```
s = b * e - c * d;
t = b * d - a * e;
det = a * c - b * b;
if ( s + t <= det )
{
    if ( s < 0 )
    {
        if ( t < 0 )
        {
            region 4
        }
        else
        {
            region 3
        }
    }
    else if ( t < 0 )
    {
        region 5
    }
    else
    {
        region 0
    }
}
else
{
    if ( s < 0 )
    {
        region 2
    }
    else if ( t < 0 )
    {
        region 6
    }
    else
    {
        region 1
    }
}
```

The block of code for handling region 0 is shown in Listing 2.

Listing 2. The source code when $(\bar{s}, \bar{t})$ is in region 0.

```
s /= det;
t /= det;
```

The block of code for handling region 1 is shown in Listing 3.

Listing 3. The source code when $(\bar{s}, \bar{t})$ is in region 1. The minimum distance must occur on the line $s + t = 1$. The elliptical level curve touches the line for some real number $\hat{s}$. If $\hat{s} \leq 0$, the minimum distance occurs at $s = 0$; if $\hat{s} \geq 1$, the minimum distance occurs at $s = 1$; otherwise, the minimum distance occurs at $s = \hat{s}$.

```
// F(s) = Q(s, 1 - s) = (a - 2b + c)s^2 + 2((b + d) - (c + e))s + (c + 2e + f)
// F'(s)/2 = (a - 2b + c)s + ((b + d) - (c + e))
// F'(ŝ) = 0 when ŝ = ((c + e) - (b + d))/(a - 2b + c)
// a - 2b + c > 0, so only the sign of (c + e) - (b + d) must be analyzed
numer = (c + e) - (b + d);
if ( numer <= 0 )
{
    s = 0;
}
else
{
    denom = a - 2 * b + c;
    if ( numer >= denom )
    {
        s = 1;
    }
    else
    {
        s = numer / denom;
    }
}
t = 1 - s;
```

The block of code for handling region 3 is shown in Listing 4.

Listing 4. The source code when $(\bar{s}, \bar{t})$ is in region 3. The minimum distance must occur on the line $s = 0$. The elliptical level curve touches the line for some real number $\hat{t}$. If $\hat{t} \leq 0$, the minimum distance occurs at $t = 0$; if $\hat{t} \geq 1$, the minimum distance occurs at $t = 1$; otherwise, the minimum distance occurs at $t = \hat{t}$.

```
// F(t) = Q(0, t) = ct^2 + 2et + f
// F'(t)/2 = ct + e
// F'(t̂) = 0 when t̂ = -e/c
s = 0;
if ( e >= 0 )
{
    t = 0;
}
else if ( -e >= c )
{
    t = 1;
}
else
{
    t = -e / c;
}
```

The block of code for handling region 5 is shown in Listing 5.

Listing 5. The source code when $(\bar{s}, \bar{t})$ is in region 5. The minimum distance must occur on the line $t = 0$. The elliptical level curve touches the line for some real number $\hat{s}$. If $\hat{s} \leq 0$, the minimum distance occurs at $s = 0$; if $\hat{s} \geq 1$, the minimum distance occurs at $s = 1$; otherwise, the minimum distance occurs at $s = \hat{s}$.

```
// F(s) = Q(s,0) = as^2 + 2ds + f
// F'(s)/2 = as + d
// F'(ŝ) = 0 when ŝ = -d/a
t = 0;
if ( d >= 0 )
{
    s = 0;
}
else if ( -d >= a )
{
    s = 1;
}
else
{
    s = -d / a;
}
```

The block of code for handling region 2 is shown in Listing 6.

Listing 6. The source code when $(\bar{s}, \bar{t})$ is in region 2. The minimum distance must occur on the line $s + t = 1$ when $(1, -1) \cdot \nabla Q(0, 1) < 0$ and the elliptical level curve touches the line for some real number $\hat{s} > 0$. If $\hat{s} \geq 1$, the minimum distance occurs at $s = 1$; otherwise, the minimum distance occurs at $s = \hat{s}$. When $(0, -1) \cdot \nabla Q(0, 1) \leq 0$, the minimum distance must occur on the line $s = 0$ with $t \leq 1$.

```
// ∇Q(s,t)/2 = (as + bt + d, bs + ct + e)
// (1,−1)·∇Q(0,1)/2 = (1,−1)·(b + d, c + e) = (b + d) − (c + e)
// (0,−1)·∇Q(0,1)/2 = (0,−1)·(b + d, c + e) = −(c + e)
// minimum on edge s + t = 1 if (1,−1)·∇Q(0,1) < 0; otherwise, minimum on edge s = 0
tmp0 = b + d;
tmp1 = c + e;
if ( tmp1 > tmp0 )  // minimum on edge s + t = 1 with s > 0
{
    // F(s) = Q(s,1 − s) = (a − 2b + c)s^2 + 2((b + d) − (c + e))s + (c + 2e + f)
    // F'(s)/2 = (a − 2b + c)s + ((b + d) − (c + e))
    // F'(ŝ) = 0 when ŝ = ((c + e) − (b + d))/(a − 2b + c)
    // a − 2b + c > 0, so only the sign of (c + e) − (b + d) must be analyzed
    numer = tmp1 - tmp0;
    denom = a - 2 * b + c;
    if ( numer >= denom )
    {
        s = 1;
    }
    else
    {
        s = numer / denom;
    }
    t = 1 - s;
}
else  // minimum on edge s = 0 with t <= 1
{
    // F(t) = Q(0,t) = ct^2 + 2et + f
    // F'(t)/2 = ct + e
    // F'(t̂) = 0 when t̂ = −e/c
    s = 0;
    if ( tmp1 <= 0 )
    {
        t = 1;
    }
    else if ( e >= 0 )
    {
        t = 0;
    }
    else
    {
        t = -e / c;
    }
}
```

The block of code for handling region 6 is shown in Listing 7.

Listing 7. The source code when $(\bar{s}, \bar{t})$ is in region 6. The minimum distance must occur on the line $s + t = 1$ when $(-1, 1) \cdot \nabla Q(1, 0) < 0$ and the elliptical level curve touches the line for some real number $\hat{t} > 0$. If $\hat{t} \geq 1$, the minimum distance occurs at $t = 1$; otherwise, the minimum distance occurs at $t = \hat{t}$. When $(-1, 0) \cdot \nabla Q(1, 0) \leq 0$, the minimum distance must occur on the line $t = 0$ with $s \leq 1$.

```
// ∇Q(s,t)/2 = (as + bt + d, bs + ct + e)
// (−1,1) · ∇Q(1,0)/2 = (−1,1) · (a + d, b + e) = (b + e) − (a + d)
// (−1,0) · ∇Q(1,0)/2 = (−1,0) · (a + d, b + e) = −(a + d)
// minimum on edge s + t = 1 if (−1,1) · ∇Q(1,0) < 0; otherwise, minimum on edge t = 0
tmp0 = b + e;
tmp1 = a + d;
if ( tmp1 > tmp0 )  // minimum on edge s + t = 1 with t > 0
{
    // F(t) = Q(1 − t, t) = (a − 2b + c)t^2 + 2((b + e) − (a + d))t + (a + 2d + f)
    // F'(t)/2 = (a − 2b + c)t + ((b + e) − (a + d))
    // F'(t̂) = 0 when t̂ = ((a + d) − (b + e))/(a − 2b + c)
    // a − 2b + c > 0, so only the sign of (a + d) − (b + e) must be analyzed
    numer = tmp1 - tmp0;
    denom = a - 2 * b + c;
    if ( numer >= denom )
    {
        t = 1;
    }
    else
    {
        t = numer / denom;
    }
    s = 1 - t;
}
else  // minimum on edge t = 0 with s <= 1
{
    // F(s) = Q(s,0) = as^2 + 2ds + f
    // F'(s)/2 = as + d
    // F'(ŝ) = 0 when ŝ = −d/a
    t = 0;
    if ( tmp1 <= 0 )
    {
        s = 1;
    }
    else if ( d >= 0 )
    {
        s = 0;
    }
    else
    {
        s = -d / a;
    }
}
```

The block of code for handling region 4 is shown in Listing 8.

Listing 8. The source code when $(\bar{s}, \bar{t})$ is in region 4. The minimum distance must occur on the line $t = 0$ when $(1, 0) \cdot \nabla Q(0, 0) < 0$ and the elliptical level curve touches the line for some real number $\hat{s} > 0$. If $\hat{s} \geq 1$, the minimum distance occurs at $s = 1$; otherwise, the minimum distance occurs at $s = \hat{s}$. When $(0, 1) \cdot \nabla Q(0, 0) \leq 0$, the minimum distance must occur on the line $s = 0$ with $t \geq 0$.

```
// ∇Q(s,t)/2 = (as + bt + d, bs + ct + e)
// (1,0) · ∇Q(0,0)/2 = (1,0) · (d,e) = d
// (0,1) · ∇Q(0,0)/2 = (0,1) · (d,e) = e
// minimum on edge t = 0 if (1,0) · ∇Q(0,0) < 0; otherwise, minimum on edge s = 0
if ( d < 0 )  // minimum on edge t = 0 with s > 0
{
    // F(s) = Q(s,0) = as^2 + 2ds + f
    // F'(s)/2 = as + d
    // F'(ŝ) = 0 when ŝ = -d/a
    t = 0;
    if ( -d >= a )
    {
        s = 1;
    }
    else
    {
        s = -d / a;
    }
}
else  // minimum on edge s = 0
{
    // F(t) = Q(0,t) = ct^2 + 2et + f
    // F'(t)/2 = ct + e
    // F'(t̂) = 0 when t̂ = -e/c
    s = 0;
    if ( e >= 0 )
    {
        t = 0;
    }
    else if ( -e >= c )
    {
        t = 1;
    }
    else
    {
        t = -e / c;
    }
}
```

The direction implementation for the point-triangle distance in 3D is DistPointTriangleExact.h. An implementation designed to be more robust to floating-point rounding errors when the triangle edges are nearly parallel is DistPointTriangle.h. Both implementations actually allow for a problem formulated in n dimensions; that is, the point and triangle can live in n dimensions for any $n \geq 2$.