Inspheres and Circumspheres of Simplices

Gautam Manohar

17 February 2018

This document originally appeared as a blog post on my website. Find it at gautammanohar.com/simplex-sphere.

Every triangle can be circumscribed by a circle, because a circle can always be drawn through three given points, so long as they are not colinear. In fact, every triangle also has a circle that is inscribed in it. These two circles are called the triangle's circumcircle and incircle. Finding these circles for a triangle is simple if the triangle is equilateral, a bit harder if it is right-angled, and a bit harder still if it is not.

What is less well-known is that every tetrahedron also has a circumsphere and insphere. Given four points, such that no three of them are colinear and not all four of them are coplanar, there is a unique sphere that goes through all of them. Finding these spheres for a regular tetrahedron is rather straightforward. In particular, a regular tetrahedron's circumsphere is always three times larger in radius than its insphere. But the analagous problem for an arbitrary, not necessarily regular, tetrahedron is much hairier. I will present a geometric proof for the case of the regular tetrahedron, and then a look into the circumsphere and insphere of arbitrary tetrahedra using techniques from linear algebra. Finally, I will generalize my work to arbitrary simplices (the higher-dimensional analogue of tetrahedra) in arbitrary dimensions.

1 Regular Tetrahedra

We wish to find the ratio between the circumsphere and insphere of a regular tetrahedron. Let $ABCD$ be a regular tetrahedron. Let O be the centroid of $ABCD$. Then because $ABCD$ is regular, O is the incenter and circumcenter of $ABCD$. Let X be the centroid of the face opposite A. The volume of $OBCD$ is a quarter that of $ABCD$, because $ABCD$ is the union of $OABC, OABD, OACD, OBCD$, and each of those tetrahedra with vertex O is congruent to the others. The volume of a tetrahedron is $\frac{1}{3}bh$. Since $ABCD$ and $OBCD$ have the same base, triangle BCD, the height OX is one quarter the height AX. But because $AO = AX - OX$, and $\frac{1}{4}|AX| = |OX|$,

then $|AO| = \frac{3}{4}|AX|$. Note that AO is the circumradius and OX is the inradius. Thus the ratio of the circumradius to the inradius in a regular tetrahedron is $\frac{|AO|}{|OX|} = 3$.

2 Irregular Tetrahedra

We use coordinate geometry and linear algebra to find the inradius and circumradius (and incenter and circumcenter) of a tetrahedron given the coordinates of its vertices. To avoid immense computational tedium, we use programming to numerically solve for the desired ratio.

2.1 Insphere

Let's first deal with the insphere. Let (a, b, c) be the vector that gives the coordinates of the incenter, and let r be the inradius. Let the four vertices of the tetrahedron be given by $\mathbf{V}_i = (x_i, y_i, z_i)$, where $i \in \{1, 2, 3, 4\}$. Let F_1 be the face with vertices $\mathbf{V}_1, \mathbf{V}_2, \mathbf{V}_3$. Let F_2 be formed by $\mathbf{V}_1, \mathbf{V}_2, \mathbf{V}_4$, F_3 by $\mathbf{V}_1, \mathbf{V}_3, \mathbf{V}_4$, and F_4 by $\mathbf{V}_2, \mathbf{V}_3, \mathbf{V}_4$. Let $\mathbf{n}_i$ represent the unit vector pointing out of the tetrahedron and perpendicular to the face F_i of the tetrahedron: that is, the unit outward-facing normal vector to F_i. The defining characteristic of the insphere is that each line from the incenter to the point of tangency on a given face is perpendicular to that face and of equal length to the other such lines. Taking a circular cross-section of the insphere such that the sphere and the circle have the same center and point of tangency to the face, we see that the tangent line to the circle must be at a right angle to the line joining the point of tangency and the center (from a famous theorem about circles). Then we have the condition that $(a, b, c) + r\mathbf{n}_i$ lies on F_i.

Let us deal with F_1. A vector perpendicular to the plane containing two vectors is given by their cross product. We can let these two vectors be two of the edges of F_1. An edge on a face is the displacement between two vertices of a face. Then the cross product

$$(\mathbf{V}_2 - \mathbf{V}_1) \times (\mathbf{V}_3 - \mathbf{V}_1) \tag{1}$$

is perpendicular to F_1. But (1) need not point outward. To remove this ambiguity, we scale (1) by a negative value if it is pointing inward and by a positive value if it is pointing outward. Consider the dot product of (1) and the displacement vector $\mathbf{V}_4 - \mathbf{V}_1$, which points from the first vertex away from the outward-facing normal and to the fourth vertex. Thus the angle between the outward-facing normal and this displacement vector is between $\frac{\pi}{2}$ and π, so the cosine of the angle is negative, so the dot product of these two vectors is negative. Then scaling (1) by the negative of the dot product yields an outward-facing normal.

We then need to normalize this vector to obtain the desired unit outward-facing normal vector $\mathbf{n}_1$:

$$\mathbf{n}_1 = -\frac{(\mathbf{V}_2 - \mathbf{V}_1) \times (\mathbf{V}_3 - \mathbf{V}_1)(\mathbf{V}_2 - \mathbf{V}_1) \times (\mathbf{V}_3 - \mathbf{V}_1) \cdot (\mathbf{V}_4 - \mathbf{V}_1)}{\|(\mathbf{V}_2 - \mathbf{V}_1) \times (\mathbf{V}_3 - \mathbf{V}_1)(\mathbf{V}_2 - \mathbf{V}_1) \times (\mathbf{V}_3 - \mathbf{V}_1) \cdot (\mathbf{V}_4 - \mathbf{V}_1)\|}. \quad (2)$$

The point-normal form for the equation of a plane perpendicular to a vector $\mathbf{n}$ and which contains a point P_0 is, for some point P on the plane,

$$(P - P_0) \cdot \mathbf{n} = 0. \quad (3)$$

In particular, we want the equation for the plane which contains F_1. We use $P = (a, b, c) + r\mathbf{n}_1$ (the point on F_1 tangent to the insphere) and $P_0 = \mathbf{V}_1$ (it can be any of the vertices of the face in question, in this case F_1). This gives

$$((a, b, c) + r\mathbf{n}_1 - \mathbf{V}_1) \cdot \mathbf{n}_1 = 0. \quad (4)$$

Distributing the dot product, we have

$$(a, b, c) \cdot \mathbf{n}_1 + r\mathbf{n}_1 \cdot \mathbf{n}_1 - \mathbf{V}_1 \cdot \mathbf{n}_1 = 0. \quad (5)$$

We then break each vector into its three components:

$$\begin{aligned} a\mathbf{n}_{1_x} + b\mathbf{n}_{1_y} + c\mathbf{n}_{1_z} + r(\mathbf{n}_{1_x}^2 + \mathbf{n}_{1_y}^2 + \mathbf{n}_{1_z}^2) & \\ -x_1\mathbf{n}_{1_x} - y_1\mathbf{n}_{1_y} - z_1\mathbf{n}_{1_z} = 0 & \quad (6) \\ a\mathbf{n}_{1_x} + b\mathbf{n}_{1_y} + c\mathbf{n}_{1_z} + r(\mathbf{n}_{1_x}^2 + \mathbf{n}_{1_y}^2 + \mathbf{n}_{1_z}^2) = x_1\mathbf{n}_{1_x} + y_1\mathbf{n}_{1_y} + z_1\mathbf{n}_{1_z}. & \end{aligned}$$

This equation has four unknowns: the coordinates of the incenter a, b, c and the inradius r. Using the other three faces, we arrive at a system of four equations with four unknowns:

$$\begin{aligned} a\mathbf{n}_{1_x} + b\mathbf{n}_{1_y} + c\mathbf{n}_{1_y} + r(\mathbf{n}_{1_x}^2 + \mathbf{n}_{1_y}^2 + \mathbf{n}_{1_z}^2) &= x_1\mathbf{n}_{1_x} + y_1\mathbf{n}_{1_y} + z_1\mathbf{n}_{1_z} \\ a\mathbf{n}_{2_x} + b\mathbf{n}_{2_y} + c\mathbf{n}_{2_z} + r(\mathbf{n}_{2_x}^2 + \mathbf{n}_{2_y}^2 + \mathbf{n}_{2_z}^2) &= x_2\mathbf{n}_{2_x} + y_2\mathbf{n}_{2_y} + z_2\mathbf{n}_{2_z} \\ a\mathbf{n}_{3_x} + b\mathbf{n}_{3_y} + c\mathbf{n}_{3_z} + r(\mathbf{n}_{3_x}^2 + \mathbf{n}_{3_y}^2 + \mathbf{n}_{3_z}^2) &= x_3\mathbf{n}_{3_x} + y_3\mathbf{n}_{3_y} + z_3\mathbf{n}_{3_z} \\ a\mathbf{n}_{4_x} + b\mathbf{n}_{4_y} + c\mathbf{n}_{4_z} + r(\mathbf{n}_{4_x}^2 + \mathbf{n}_{4_y}^2 + \mathbf{n}_{4_z}^2) &= x_4\mathbf{n}_{4_x} + y_4\mathbf{n}_{4_y} + z_4\mathbf{n}_{4_z} \end{aligned} \quad (7)$$

We can solve this system with matrices. Let

$$\mathbf{A} = \begin{bmatrix} \mathbf{n}_{1_x} & \mathbf{n}_{1_y} & \mathbf{n}_{1_y} & \mathbf{n}_{1_x}^2 + \mathbf{n}_{1_y}^2 + \mathbf{n}_{1_z}^2 \\ \mathbf{n}_{2_x} & \mathbf{n}_{2_y} & \mathbf{n}_{2_y} & \mathbf{n}_{2_x}^2 + \mathbf{n}_{2_y}^2 + \mathbf{n}_{2_z}^2 \\ \mathbf{n}_{3_x} & \mathbf{n}_{3_y} & \mathbf{n}_{3_y} & \mathbf{n}_{3_x}^2 + \mathbf{n}_{3_y}^2 + \mathbf{n}_{3_z}^2 \\ \mathbf{n}_{4_x} & \mathbf{n}_{4_y} & \mathbf{n}_{4_y} & \mathbf{n}_{4_x}^2 + \mathbf{n}_{4_y}^2 + \mathbf{n}_{4_z}^2 \end{bmatrix} \quad (8)$$

Let

$$\mathbf{u} = \begin{bmatrix} x_1\mathbf{n}_{1_x} + y_1\mathbf{n}_{1_y} + z_1\mathbf{n}_{1_z} \\ x_2\mathbf{n}_{2_x} + y_2\mathbf{n}_{2_y} + z_2\mathbf{n}_{2_z} \\ x_3\mathbf{n}_{3_x} + y_3\mathbf{n}_{3_y} + z_3\mathbf{n}_{3_z} \\ x_4\mathbf{n}_{4_x} + y_4\mathbf{n}_{4_y} + z_4\mathbf{n}_{4_z} \end{bmatrix} \quad (9)$$

And let the solution vector be

$$\mathbf{v} = \begin{bmatrix} a \\ b \\ c \\ r \end{bmatrix} \tag{10}$$

Then

$$\begin{aligned} \mathbf{A}\mathbf{v} &= \mathbf{u} \\ \mathbf{v} &= \mathbf{A}^{-1}\mathbf{u}. \end{aligned} \tag{11}$$

And so the inradius r is the fourth element of $\mathbf{v}$, or $\mathbf{v}_4$.

2.2 Circumsphere

Now we deal with the circumsphere. Let the circumsphere have center (A, B, C) and radius R. It then has the equation

$$(x - A)^2 + (y - B)^2 + (z - C)^2 = R^2 \tag{12}$$

All four of a tetrahedron's vertices lie on its circumsphere, and so their coordinates satisfy (12). We have four vertices with which we can make a system of four equations:

$$\begin{aligned} (x_1 - A)^2 + (y_1 - B)^2 + (z_1 - C)^2 &= R^2 \\ (x_2 - A)^2 + (y_2 - B)^2 + (z_2 - C)^2 &= R^2 \\ (x_3 - A)^2 + (y_3 - B)^2 + (z_3 - C)^2 &= R^2 \\ (x_4 - A)^2 + (y_4 - B)^2 + (z_4 - C)^2 &= R^2 \end{aligned} \tag{13}$$

Subtracting the first equation from each of the other three, we have another system of equations:

$$\begin{aligned} 2A(x_1 - x_2) + 2B(y_1 - y_2) + 2C(z_1 - z_2) &= x_1^2 + y_1^2 + z_1^2 - x_2^2 - y_2^2 - z_2^2 \\ 2A(x_1 - x_3) + 2B(y_1 - y_3) + 2C(z_1 - z_3) &= x_1^2 + y_1^2 + z_1^2 - x_3^2 - y_3^2 - z_3^2 \\ 2A(x_1 - x_4) + 2B(y_1 - y_4) + 2C(z_1 - z_4) &= x_1^2 + y_1^2 + z_1^2 - x_4^2 - y_4^2 - z_4^2 \end{aligned} \tag{14}$$

We can solve this system of equations using matrices. Let

$$\mathbf{M} = 2\begin{bmatrix} x_1 - x_2 & y_1 - y_2 & z_1 - z_2 \\ x_1 - x_3 & y_1 - y_3 & z_1 - z_3 \\ x_1 - x_4 & y_1 - y_4 & z_1 - z_4 \end{bmatrix} \tag{15}$$

Let

$$\mathbf{u} = \begin{bmatrix} x_1^2 + y_1^2 + z_1^2 - x_2^2 - y_2^2 - z_2^2 \\ x_1^2 + y_1^2 + z_1^2 - x_3^2 - y_3^2 - z_3^2 \\ x_1^2 + y_1^2 + z_1^2 - x_4^2 - y_4^2 - z_4^2 \end{bmatrix} \tag{16}$$

And let the solution vector be

$$\mathbf{v} = \begin{bmatrix} A \\ B \\ C \end{bmatrix} \tag{17}$$

Then

$$\mathbf{v} = \mathbf{M}^{-1}\mathbf{u}. \tag{18}$$

Having solved for A, B, C, we can solve for R from one of the equations in (13). Let's use the first one.

$$\begin{aligned} R^2 &= (x_1 - A)^2 + (y_1 - B)^2 + (z_1 - C)^2 \\ R &= \sqrt{(x_1 - A)^2 + (y_1 - B)^2 + (z_1 - C)^2}. \end{aligned} \tag{19}$$

Although we could compute the components of $\mathbf{n}_i$ by doing long and tedious calculations with cross products, dot products, and norms and invert our matrices using, say, Gaussian elimination, the closed form expressions for the inradius r and the circumradius R would be very unwieldy and so not very useful.

3 Generalizing to n-Simplices

Our method generalizes readily to higher dimensions. The dot product generalizes very simply. The one thing we need is a generalization of the cross product, which is only defined in three dimensions... except we don't. Our only use for the cross product was to find a vector normal to two others. The cross product can be represented as the determinant of a matrix

$$\begin{vmatrix} \mathbf{v}_1^1 & \mathbf{v}_1^2 & \mathbf{v}_1^3 \\ \mathbf{v}_2^1 & \mathbf{v}_2^2 & \mathbf{v}_2^3 \\ \mathbf{b}_1 & \mathbf{b}_2 & \mathbf{b}_3 \end{vmatrix} \tag{20}$$

Here the subscript is the index of the vector (out of those we are taking the cross product of) and the superscript is the index of the component in that vector. Each vector $\mathbf{b}_i$ is the i-th basis vector of our n-dimensional vector space. We can then write our generalized cross product of $n-1$ vectors in n-dimensions as

$$\mathcal{C}(\mathbf{v}_1, \mathbf{v}_2, \ldots, \mathbf{v}_{n-1}) = \begin{vmatrix} \mathbf{v}_1^1 & \cdots & \mathbf{v}_1^n \\ \vdots & \ddots & \vdots \\ \mathbf{v}_{n-1}^1 & \cdots & \mathbf{v}_{n-1}^n \\ \mathbf{b}_1 & \cdots & \mathbf{b}_n \end{vmatrix} \tag{21}$$

But we can get even simpler. Determinants can get hard to compute as they get large. And because we are only interested in the cross product for its *direction* (and not its magnitude; we normalize everything later anyway), we can use a simpler method. Any element of the kernel of the $n-1$ by n matrix given by

taking the $n-1$ vectors to be rows is in the orthogonal direction to all the given vectors.

An n simplex is given by $n+1$ points, given by the vectors $\mathbf{V}_i$, where i is an integer from 1 to $n+1$. Each of these vectors has n components, $\mathbf{V}_i^j$, where j is an integer from 1 to n. Each of the $n+1$ faces is a collection of n of the $n+1$ points. Again, let's first deal with the insphere:

3.1 Insphere

Let the vector representing the insphere's coordinates have n components c_i. Imitating (2), we have

$$\mathbf{n}_1 = -\frac{\mathbf{C}(\mathbf{C}\cdot(\mathbf{V}_{n+1}-\mathbf{V}_1))}{\|\mathbf{C}(\mathbf{C}\cdot(\mathbf{V}_{n+1}-\mathbf{V}_1))\|}, \tag{22}$$

where

$$\mathbf{C} = \mathcal{C}((\mathbf{V}_2-\mathbf{V}_1)\times(\mathbf{V}_3-\mathbf{V}_1)\times\cdots\times(\mathbf{V}_n-\mathbf{V}_1)). \tag{23}$$

Then we can use matrices again:

$$\mathbf{A} = \begin{bmatrix}\mathbf{n}_i^j & \sum(\mathbf{n}_i^j)^2\end{bmatrix} \tag{24}$$

$$\mathbf{u} = \begin{bmatrix}\sum \mathbf{V}_i^j\mathbf{n}_i^j\end{bmatrix} \tag{25}$$

$$\mathbf{s} = \begin{bmatrix} c_1 \\ c_2 \\ \vdots \\ c_n \\ r \end{bmatrix} \tag{26}$$

Then

$$\mathbf{As} = \mathbf{u}. \tag{27}$$

3.2 Circumsphere

Let the vector representing the insphere's coordinates have n components C_i. Our procedure for finding the circumsphere generalizes readily as follows.

$$\mathbf{M} = 2\begin{bmatrix} \mathbf{V}_1^1-\mathbf{V}_2^1 & \cdots & \mathbf{V}_1^n-\mathbf{V}_2^n \\ \vdots & \ddots & \vdots \\ \mathbf{V}_1^1-\mathbf{V}_{n+1}^1 & \cdots & \mathbf{V}_1^n-\mathbf{V}_{n+1}^n \end{bmatrix} \tag{28}$$

$$\mathbf{u} = \begin{bmatrix} \sum((v_1^j)^2-(v_2^j)^2) \\ \vdots \\ \sum((v_1^j)^2-(v_{n+1}^j)^2) \end{bmatrix} \tag{29}$$

$$\mathbf{s} = \begin{bmatrix} C_1 \\ C_2 \\ \vdots \\ C_n \\ r \end{bmatrix} \tag{30}$$

Then

$$\mathbf{As} = \mathbf{u}. \tag{31}$$

3.3 Implementing Our Higher Dimensional Processes

We'll use `Python` and its `numpy` package. Let's first set up a function to compute our generalized cross product using the kernel method. Here's a standard implementation of finding the kernel, or nullspace, of a matrix, sourced from here.

```
def nullspace(A, atol=1e-13, rtol=0):
    A = np.atleast_2d(A)
    u, s, vh = np.linalg.svd(A)
    tol = max(atol, rtol * s[0])
    nnz = (s >= tol).sum()
    ns = vh[nnz:].conj().T
    return np.array([k[0] for k in ns])
```

We return the element of the nullspace that is a vector of norm 1 and use it to calculate the generalized cross product.

```
def gen_cross(v):
    n = len(v)
    m = np.array([[v[i][j] for j in range(n+1)] for i in range(↩
        n)])
    return nullspace(m)
```

The argument `v` will always be a list of lists; each element represents a vector. Now we can calculate the incenter and inradius:

```
def insphere(v):
    n = len(v) - 1
    faces = []
    normals = []
```

Note that `n` is the "dimension" of the tetrahedron.

```
for i in range(n+1):
```

```
        faces.append(v + [v[n-i]])
        del faces[i][n-i]
    if n % 2 == 0:
        faces = list(reversed(faces))
    faces = np.array(faces)
```

Each face of an n-simplex consists of some n of the given $n+1$ vertices. Each element of `faces` has $n+1$ elements: the n vertices of the face, and the other vertex that is not part of the face. The statement involving the modulo ensures that the orientation is correct in even dimensions. Then we can finish it off by performing the calculations in (22) and setting up the matrices as in (24) and (25).

```
    for f in faces:
        c = gen_cross([f[i] - f[0] for i in range(1,n)])
        unnormed = c * np.vdot(c, f[n] - f[0])
        normals.append((-1)**(n%2) * unnormed / np.linalg.norm(↩
            unnormed))
    A = np.array([[k[i] for i in range(n)] + [sum([k[i]**2 for ↩
        i in range(n)])] for k in normals])
    B = np.array([sum([v[i][j]*normals[i][j] for j in range(n)↩
        ]) for i in range(n+1)])
    return np.linalg.solve(A,B)
```

We go along a similar line for the circumsphere: we set up the matrices and solve.

```
def circumsphere(v):
    n = len(v) - 1
    A = 2 * np.array([[v[0][j] - v[i][j] for j in range(n)] for↩
         i in range(1,n+1)])
    B = np.array([sum([v[0][j]**2 - v[i][j]**2 for j in range(n↩
        )]) for i in range(1,n+1)])
    sols = np.linalg.solve(A,B)
    R = sqrt(sum([(v[0][i] - sols[i])**2 for i in range(n)]))
    return np.append(sols, R)
```

And there you have it! It would be interesting to see how the ratio between circumsphere and insphere (3 in 3 dimensions) changes as the dimension increases. It will probably get much larger (because most of a high-dimensional sphere's volume is near its surface and because the insphere's volume will become very small), but how quickly?