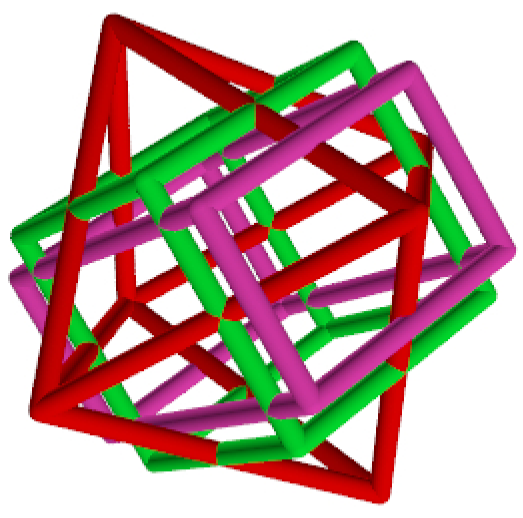
The Rootful Cuboid



A round of applause for this David Koski discovery, perhaps known to nomads, but missing from contemporary popularizations:

The cuboid of height, width, depth $\sqrt{1}$ $\sqrt{2}$ $\sqrt{4}$, and face diagonals $\sqrt{3}$ $\sqrt{5}$ $\sqrt{6}$ and body diagonal $\sqrt{7}$.

That's a parellelopiped with an inscribed tetrahedron of only face diagonals, and four right tetrahedrons each involving a corner and three right angles.

The inscribed tetrahedron will have a volume 1/3rd that of the paralleopiped, with the corner tetrahedrons each having 1/6th that total volume. This is a generic truth for parallelopipeds.

Lets label the two ends of the cuboic (brick) ABCD counter-clockwise with EFGH at the other end, and matching A with E, B with F and so on, such that the six faces of the cuboid are:

- ABCD, EFGH (opposite front/back)
- AEFB, DHGF (parallel sides)
- ADHE, BCGF (parallel top/bottom)

The convention for naming edges is to write the two corner points alphabetically, to keep them unique. We write AB, but not BA, as these describe the same edge and we want a unique canonical identifier for each one.

In [1]:

```
from math import sqrt as rt2

# cuboid
AB = EF = CD = GH = rt2(1)/2
AD = BC = FG = EH = rt2(2)/2
AC = BD = EG = FH = rt2(3)/2
AE = BF = CG = DH = rt2(4)/2
AF = BE = CH = DG = rt2(5)/2
AH = BG = CF = DE = rt2(6)/2
AG = BH = CE = DF = rt2(7)/2
```

Enter our frame of reference made of unit radius spheres packed in the CCP interconnected by IVM vectors. Four such spheres packed together define a tetrahedron of edges one sphere diameter (D), or twice the radius (R).

If we wish our home base tetrahedron of edges one, to have volume one, then set we set our unit length to D (= 2 R).

The reason for scaling the entire cuboid describe above by 1/2 is to imagine we're going from R as unit length to D as unit length. Every length gets halved, and the result is an inscribed tetradron of unit tetravolume, the same as our reference tetrahedron's, even though this one is irregular.

The transformation from XYZ volume to IVM volume is baked into the constant S3, such that we sometimes say "XYZ volume times S3 gives IVM volume" with 1/S3 used to go the other direction. S3 = $\sqrt{9/8}$ or $(3\sqrt{2})/4$.

In [2]:

```
import tetravolume as tv
```

The volume method used below takes the six lengths of a tetrahedron as input.

Order matters.

Pick a corner, any corner, of the tetrahedron, and start with the three lengths stemming from that corner, to the corners of the opposite face.

Then provide the segments around the opposite face, in the same order mentioned.

For example, given tetrahedron ABCD, lets pick A and go AB, AC, AD.

The opposite face is BCD and the lengths should be in the order mentioned the first time i.e. B first, so B-to-C, C-to-D and D-back-to-B i.e. BC, CD, BD.

Remember how edges have a unique representation and we write BD versus DB, because B is before D alphabetically.

In [3]:

```
unit_vol = tv.Tetrahedron(1, 1, 1, 1, 1, 1) # reference tetrahedron
```

In [4]:

```
unit_vol.ivm_volume()
```

```
Out[4]: 1.0
```

In [5]:

```
ACHF = tv.Tetrahedron(AC, AH, AF, CH, FH, CF) # inscribed tetrahedron, all face diago
ACHF.ivm_volume()
```

```
Out[5]: 1.0
```

In [6]:

```
ACHF.xyz_volume()
```

```
Out[6]: 0.9428090415820635
```

In [7]:

```
AEDB = tv.Tetrahedron(AB, AE, AD, BE, DE, BD) # right angles at A
AEDB.ivm_volume()
```

```
Out[7]: 0.5
```

In [8]:

```
ABCE = tv.Tetrahedron(AB, AC, AE, BC, CE, BE) # right angles at B
ABCE.ivm_volume()
```

```
Out[8]: 0.49999999999999956
```

In [9]:

```
ACDH = tv.Tetrahedron(AD, CD, DH, AC, CH, AH)  # right angles at H
ACDH.ivm_volume()
```

```
Out[9]: 0.5
```

In [10]:

```
FGHC = tv.Tetrahedron(FG, GH, CG, FH, CH, CF) # right angles at G
FGHC.ivm_volume()
```

```
Out[10]: 0.5
```

In [11]:

```
ABCF = tv.Tetrahedron(AB, AC, AF, BC, CF, BF) # includes body diagonal
ABCF.ivm_volume()
```

```
Out[11]: 0.5
```

In [12]:

```
S3 = rt2(9/8)
```

In [13]:

```
1/S3 * 3
```

```
Out[13]: 2.8284271247461903
```

In [14]:

```
rt2(2)**3
```

```
Out[14]: 2.8284271247461907
```