

Imaginative solutions to the Snub Dodecahedron

- March 7 2019
- A mathematical essay by Mark Shelby Adams
- Mark's ORCID: <https://orcid.org/0000-0003-4469-051X> (<https://orcid.org/0000-0003-4469-051X>)

Part 1. Solutions to the Snub Dodecahedron

In 1954, H. S. M. Coxeter, M. S. Longuet-Higgins, and J. C. P. Miller published "Uniform Polyhedra." in the Philosophical Transactions of the Royal Society of London. The Snub Dodecahedron is elegantly solved in the single line equation:

$$X^3 + 2X^2 - \varphi^2 = 0$$

Harold Scott MacDonald Coxeter as shown in Wikipedia

Eric Wolfgang Weisstein brought forth Coxeter's solution to the Snub Dodecahedron in his MathWorld--A Wolfram Web Resource. In 2010, this closed-form expression for the volume was brought from MathWorld to Wikipedia.

$$Volume = \frac{12\xi^2(3\varphi + 1) - \xi(36\varphi + 7) - (53\varphi + 6)}{6\sqrt{3 - \xi^2}} \approx 37.6166499627333629757777$$

Mark Shelby Adams

At the 1986 International Congress of Mathematicians, Mark Adams presented his closed-form expression from his book, Archimedean & Platonic Solids. Shown in this essay in Part 2:

$$Volume = \frac{10\varphi}{3}\sqrt{\varphi^2 + 3\xi(\varphi + \xi)} + \frac{\varphi^2}{2}\sqrt{5 + 5\sqrt{5}\varphi\xi(\varphi + \xi)} \approx 37.6166499627333629757777$$

For both of the above expressions, the Golden Ratio Phi and Xi are defined as:

$$\varphi \equiv \frac{1 + \sqrt{5}}{2} \quad \xi \equiv \sqrt[3]{\frac{\varphi}{2} + \frac{1}{2}\sqrt{\varphi - \frac{5}{27}}} + \sqrt[3]{\frac{\varphi}{2} - \frac{1}{2}\sqrt{\varphi - \frac{5}{27}}}$$

Harold Scott MacDonald Coxeter as shown in MathWorld

MathWorld currently shows the volume as Coxeter's polynomial expression:

$$187445810737515625 - 182124351550575000x^2 + 6152923794150000x^4 + 1030526618040000x^6 + 162223191936000x^8 - 3195335070720x^{10} + 2176782336x^{12} = 0 \quad x = Volume \approx 37.6166499627333629757777$$

Harish Chandra Rajpoot

Harish Chandra Rajpoot published his 2015 paper, "Optimum Solution of Snub Dodecahedron". HCR's Theory of Polygon & Newton-Raphson Method is used to calculate the volume of the Snub Dodecahedron. His numerical method is shown in Part 3. After only 7 iterations, the calculated volume matches the closed-form solutions to 50 digits of accuracy.

$$Volume \approx 37.6166499627333629757777$$

3D Numerical

Part 3. of this essay is a Python language script that calculates the volume of the Snub Dodecahedron in five different ways. The four methods above and finally a 3D Numerical method. Two triangle objects are defined as adjacent triangles on a regular icosahedron. A root finder algorithm is applied to one point on the plane of each triangle so that the distance of a side of an inscribed snub triangle is equal to the side of a non inscribed snub triangle. The results match the other methods:

$$Volume \approx 37.6166499627333629757777$$

Part 2. Solution to the Snub Dodecahedron by Mark Adams

The Snub Dodecahedron is inscribed on to a base icosahedron of unit edge length, as shown in Figure 1.

```
In [1]: from IPython.core.display import SVG
        SVG(filename='fig_1.svg')
```

Out[1]:

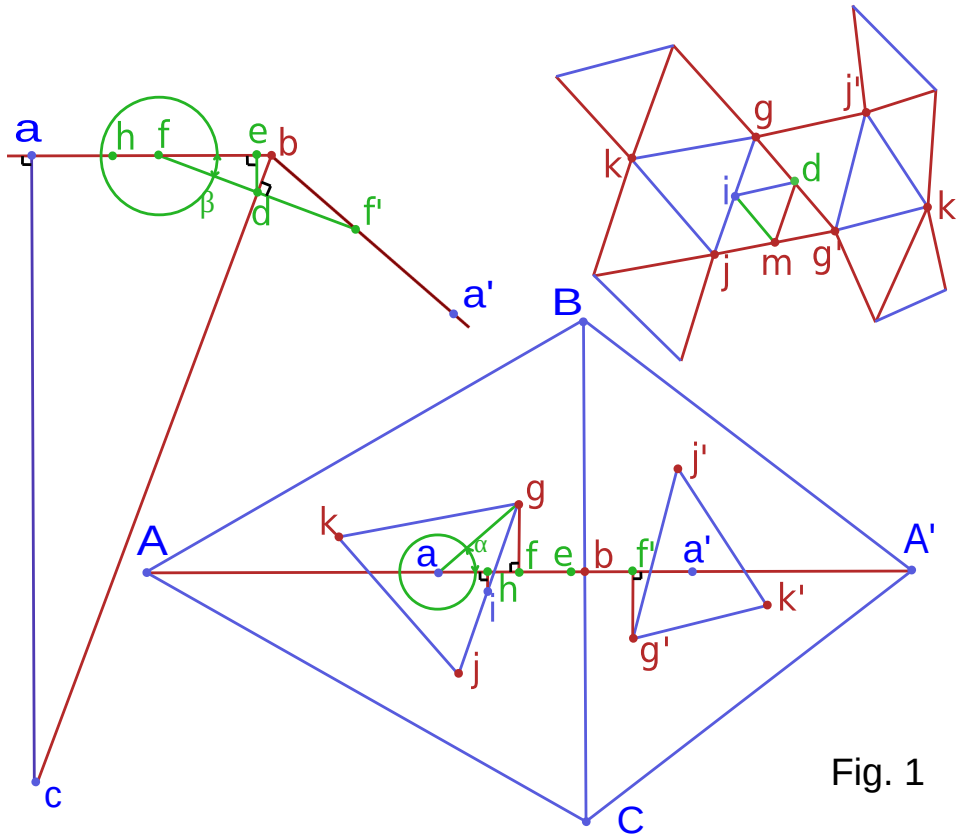


Fig. 1

Base Icosahedron faces : $\triangle ABC, \triangle A'BC$

Inscribed Snub Dodecahedron faces : $\triangle gjk, \triangle g'j'k'$

Non Inscribed Snub Dodecahedron faces : $\triangle g'gj, \triangle g'gj'$

Mid points on $\triangle g'gj$: i, d, m

Center point for both $\triangle ABC$ and $\triangle gjk$: a

Center point for both $\triangle A'BC$ and $\triangle g'j'k'$: a'

Center point for both Icosahedron and SnubDodecahedron : c

Right angles : $\angle fed, \angle fdb, \angle afg, \angle ahi, \angle af'g'$

Distance of the Snub Dodecahedron edge : D

$$\overline{AB} = \overline{BC} = \overline{CA} = \overline{A'B} = \overline{CA'} = 1$$

$$\overline{gj} = \overline{jk} = \overline{kg} = \overline{gg'} = \overline{g'j} = D = \sqrt{3} \sin \alpha - \cos \alpha$$

$$\overline{gi} = \overline{ij} = \overline{id} = \overline{gd} = \overline{dg'} = \frac{D}{2}$$

$$\overline{af}^2 + \overline{fg}^2 = \overline{ag}^2 = \frac{D^2}{3}$$

$$\overline{ah}^2 + \overline{hi}^2 = \overline{ai}^2 = \frac{D^2}{12}$$

$$\overline{ah} = \overline{ai} \cos(60 - \alpha) = \frac{D}{4\sqrt{3}} (\cos \alpha + \sqrt{3} \sin \alpha)$$

$$\overline{af} = \overline{ag} \cos \alpha = \frac{D}{\sqrt{3}} \cos \alpha$$

$$\overline{fb} = \overline{ab} - \overline{af} = \frac{1}{2\sqrt{3}} (1 - 2D \cos \alpha)$$

$$\overline{eb} = \overline{db} \sin \beta = \overline{fb} \sin^2 \beta$$

$$\overline{eb}^2 + \overline{ed}^2 = \overline{eb}^2 (1 + \cot^2 \beta) = \overline{eb} \overline{fb}$$

Equations 1 and 2

Equations 1 and 2 are second order equations of D

$$\begin{aligned}
 \overline{gd}^2 - \frac{D^2}{4} &= \overline{gf}^2 + (\overline{ab} - \overline{af} - \overline{eb})^2 + \overline{ed}^2 - \frac{D^2}{4} = 0 \\
 \overline{ag}^2 + \overline{ab}^2 - 2\overline{ab}\overline{af} + \overline{eb}[2\overline{af} - 2\overline{ab} + \overline{fb}] - \frac{D^2}{4} &= 0 \\
 \frac{D^2}{3} + \frac{1}{12} - \frac{D}{3}\cos\alpha + \sin^2\beta \frac{1 - 2D\cos\alpha}{2\sqrt{3}} \left[\frac{2D\cos\alpha}{\sqrt{3}} - \frac{1}{\sqrt{3}} + \frac{1 - 2D\cos\alpha}{2\sqrt{3}} \right] - \frac{D^2}{4} &= 0 \\
 D^2 - 4D\cos\alpha + 1 - \sin^2\beta(1 - 2D\cos\alpha)^2 &= 0 \quad (Eq.1) \\
 \overline{id}^2 - \frac{D^2}{4} &= \overline{hi}^2 + (\overline{ab} - \overline{ah} - \overline{eb})^2 + \overline{ed}^2 - \frac{D^2}{4} = 0 \\
 \overline{ai}^2 + \overline{ab}^2 - 2\overline{ab}\overline{ah} + \overline{eb}[2\overline{ah} - 2\overline{ab} + \overline{fb}] - \frac{D^2}{4} &= 0 \\
 \frac{D^2}{12} + \frac{1}{12} - \frac{D}{12}(\cos\alpha + \sqrt{3}\sin\alpha) + \sin^2\beta \frac{1 - 2D\cos\alpha}{2\sqrt{3}} \left[\frac{D(\cos\alpha + \sqrt{3}\sin\alpha)}{2\sqrt{3}} - \frac{1}{\sqrt{3}} + \frac{1 - 2D\cos\alpha}{2\sqrt{3}} \right] - \frac{D^2}{4} &= 0 \\
 -2D^2 - D(\cos\alpha + \sqrt{3}\sin\alpha) + 1 - \sin^2\beta(1 - 2D\cos\alpha)[1 + D(\cos\alpha + \sqrt{3}\sin\alpha)] &= 0 \quad (Eq.2)
 \end{aligned}$$

Equations 3 and 4

Equations 3 and 4 are trigonometric operators defining gamma letters: γ and Γ

$$\gamma \equiv \sqrt{3} \tan \alpha \quad \Gamma \equiv 3 \cos \alpha - \sqrt{3} \sin \alpha$$

$$\cos^2 \alpha + \sin^2 \alpha = 1 = \cos^2 \alpha \left(1 + \frac{\gamma^2}{3} \right) \quad \text{or} \quad \cos^2 \alpha = \frac{1}{1 + \frac{\gamma^2}{3}}$$

$$\Gamma \cos \alpha = (\cos \alpha - \sqrt{3} \sin \alpha) \cos \alpha = (3 - \gamma) \cos^2 \alpha = \frac{3 - \gamma}{1 + \frac{\gamma^2}{3}}$$

$$3 \left(1 + \frac{\gamma^2}{3} \right) \left[\Gamma \cos \alpha - \frac{3 - \gamma}{1 + \frac{\gamma^2}{3}} \right] = \overbrace{\Gamma \cos \alpha}^a \gamma^2 + \overbrace{3}^b \gamma + \overbrace{3(\Gamma \cos \alpha - 3)}^c = 0$$

$$\text{Positive Root : } \gamma = \frac{-b + \sqrt{b^2 - 4ac}}{2a} \quad \text{and} \quad \Gamma^2 = \Gamma \cos \alpha (3 - \gamma)$$

$$\gamma = \frac{-3 + \sqrt{9 - 12\Gamma \cos \alpha (\Gamma \cos \alpha - 3)}}{2\Gamma \cos \alpha} \quad (Eq.3)$$

$$\Gamma^2 = 3\Gamma \cos \alpha - \frac{1}{2} \left[-3 + \sqrt{9 - 12\Gamma \cos \alpha (\Gamma \cos \alpha - 3)} \right] \quad (Eq.4)$$

Equation 5

Combine Equations 1 and 2 with variable y to solve between D and α

Golden Ratio ϕ : $\phi \equiv \frac{1+\sqrt{5}}{2}$

Icosa symmetry: $\sin^2 \beta = \frac{1}{3\varphi^2}$ Combine: $3\varphi^2(\text{Eq.2}) + 3\varphi^2(\text{Eq.1})y = 0$

$$\overbrace{[3\varphi^2(y-2) + 2((1-2y)\cos\alpha - \sqrt{3}\sin\alpha)\cos\alpha]}^i D^2 - \overbrace{((4y+1)\cos\alpha + \sqrt{3}\sin\alpha)\varphi^4 D + (y+1)\varphi^4}^j = 0$$

Define η and λ : $j^2 - 4ik \equiv (\eta\cos\alpha + \lambda\sqrt{3}\sin\alpha)^2 = \eta^2\cos^2\alpha + (\eta\lambda)2\sqrt{3}\cos\alpha\sin\alpha + \lambda^2 3\sin^2\alpha$

$$\underbrace{[\varphi^2(4y+1)^2 - 4\varphi^2(y+1)(3\varphi^2(y-2) + 2(1-2y))]}_{\eta^2} \cos^2\alpha + \underbrace{[\varphi^8(4y+1) + 4\varphi^4(y+1)]}_{(\eta\lambda)} 2\sqrt{3}\cos\alpha\sin\alpha + \underbrace{[\varphi^8 - 4\varphi^6(y+1)(y-2)]}_{\lambda^2} 3\sin^2\alpha$$

Sum components of $(\eta\lambda)^2 - \eta^2\lambda^2 = 0$ and simplify using the identity $\varphi^{n+2} + \varphi^{n-2} = 3\varphi^n$

φ^{16}						$16 - 16$			$8 - 8$			$1 - 1$
φ^{14}		64			-32			-144			-80	144
φ^{12}		-48	144		96	0		$32 + 128$	-288		$40 - 200$	
φ^{10}		64			-32			-192			-32	
φ^8								16			32	
$\div 144\varphi^{12}$			y^4						$-2y^2$			$-\varphi^2 y$
												-144
												$8 - 184$
												144
												64
												16
												$-\varphi$

First root of y : -1

$$a = \frac{-p^2}{3} + q = \frac{-1}{3} - 1 = \frac{-4}{3} \quad (a = \frac{2}{3} \text{ plotted in Mark's book}) \quad (y+1)$$

$$b = \frac{2p^2}{27} - \frac{pq}{3} + r$$

$$b = -\frac{2}{27} - \frac{1}{3} - \varphi = \frac{-49-27\sqrt{5}}{54}$$

Second root of y :

$$\frac{-p}{3} - \sqrt{\frac{b}{2} + \sqrt{\frac{b^2}{4} + \frac{a^3}{27}}} - \sqrt[3]{\frac{b}{2} - \sqrt{\frac{b^2}{4} + \frac{a^3}{27}}} = \xi^2 - \frac{1}{3}$$

Define ξ : $\xi \equiv \sqrt[3]{\frac{\varphi}{2} + \frac{1}{2}\sqrt{\varphi - \frac{5}{27}}} + \sqrt[3]{\frac{\varphi}{2} - \frac{1}{2}\sqrt{\varphi - \frac{5}{27}}}$

From the first root of y : $\frac{(Eq.2) - (Eq.1)}{D} = 0$

$$[2(3\cos\alpha - \sqrt{3}\sin\alpha)\cos\alpha - 9\varphi^2] D + \varphi^4(3\cos\alpha - \sqrt{3}\sin\alpha) = [2\Gamma\cos\alpha - 9\varphi^2] D + \varphi^4\Gamma = 0$$

$$D = \frac{\varphi^4\Gamma}{9\varphi^2 - 2\Gamma\cos\alpha} \quad (\text{Eq.5})$$

y^3	$\overbrace{-y^2}^{p=-1}$	$\overbrace{-y}^{q=-1}$	$\overbrace{-\varphi}^{r=-\varphi}$
y^4	$-2y^2$	$-\varphi^2 y$	$-\varphi$
$+y^3$			
$-y^3$	$-y^2$		
0	$-y$		
0			
0	0	$-\varphi y$	$-\varphi$
		0	0

Equation 6

$$\frac{\Gamma(9\varphi^2 - 2\Gamma \cos \alpha)}{3\varphi^2 D} (Eq.1) = 0 \quad (\text{substitute } D \text{ with Eq.5})$$

$$\frac{\Gamma(9\varphi^2 - 2\Gamma \cos \alpha)}{3\varphi^2} \left[(3\varphi^2 - 4 \cos^2 \alpha) \left(\frac{\varphi^4 \Gamma}{9\varphi^2 - 2\Gamma \cos \alpha} \right) - 4\varphi^4 \cos \alpha + \varphi^4 \left(\frac{9\varphi^2 - 2\Gamma \cos \alpha}{\varphi^4 \Gamma} \right) \right] = 0$$

$$4(\Gamma \cos \alpha)^2 - 36\varphi^2 \Gamma \cos \alpha + 27\varphi^2 + \varphi^4 \Gamma^2 = 0 \quad (\text{substitute } \Gamma^2 \text{ with Eq.4})$$

$$4(\Gamma \cos \alpha)^2 + 3\varphi^2(\varphi^4 - 12)\Gamma \cos \alpha + \frac{3}{2}\varphi^2(\varphi^2 + 18) = \frac{\varphi^4}{2} \sqrt{9 - 12\Gamma \cos \alpha(\Gamma \cos \alpha - 3)}$$

Square both sides and subtract

$$16(\Gamma \cos \alpha)^4 + 24\varphi^2(\varphi^2 - 12)(\Gamma \cos \alpha)^3 + 36\varphi^2(21\varphi^2 + 11)(\Gamma \cos \alpha)^2 + 54\varphi^4(\varphi^2 - 36)\Gamma \cos \alpha + 81\varphi^4(\varphi^2 + 9) = 0$$

Define x : $x \equiv \frac{2}{3}\Gamma \cos \alpha$ and divide by 81:

$$x^4 + \varphi^2(\varphi^2 - 12)x^3 + \varphi^2(21\varphi^2 + 11)x^2 + \varphi^4(\varphi^2 - 36)x + \varphi^4(\varphi^2 + 9) = 0$$

$$\begin{array}{rcccccc} & & x^3 & \overbrace{-9\varphi^2}^{p=-9\varphi^2} x^2 & \overbrace{+\varphi^2(21\varphi^2+2)}^{q=\varphi^2(21\varphi^2+2)} x & \overbrace{-\varphi^4(\varphi^2+9)}^{r=-\varphi^4(\varphi^2+9)} \\ (x-1) & \overline{\begin{array}{r} x^4 \\ x^4 \\ 0 \end{array}} & +\varphi^2(\varphi^2-12)x^3 & +\varphi^2(21\varphi^2+11)x^2 & +\varphi^4(\varphi^2-36)x & +\varphi^4(\varphi^2+9) \\ & & -x^3 & & & \\ & & -9\varphi^2 x^3 & +9\varphi^2 x^2 & & \\ & & 0 & \varphi^2(21\varphi^2+2)x^2 & -\varphi^2(21\varphi^2+2)x & \\ & & & 0 & -\varphi^4(\varphi^2+9)x & \varphi^4(\varphi^2+9) \\ & & & & 0 & 0 \end{array}$$

First root of x : 1

$$a = \frac{-p^2}{3} + q = -27\varphi^4 + \varphi^2(21\varphi^2 + 2) = -2\varphi^6$$

$$b = \frac{2p^2}{27} - \frac{pq}{3} + r$$

$$b = -54\varphi^6 + 3\varphi^4(21\varphi^2 + 2) - \varphi^4(\varphi^2 + 9) = \varphi^{10}$$

Second root of x :

$$\frac{-p}{3} - \sqrt[3]{\frac{b}{2} + \sqrt{\frac{b^2}{4} + \frac{a^3}{27}}} - \sqrt[3]{\frac{b}{2} - \sqrt{\frac{b^2}{4} + \frac{a^3}{27}}} = 3\varphi^2 - \varphi^3 \xi$$

From the second root of x :

$$\Gamma \cos \alpha = \frac{3}{2}(3\varphi^2 - \varphi^3 \xi) \quad (Eq.6)$$

Equation 7

$$\left(\sqrt[3]{\frac{\varphi}{2} + \frac{1}{2}\sqrt{\varphi - \frac{5}{27}}} \right) \left(\sqrt[3]{\frac{\varphi}{2} - \frac{1}{2}\sqrt{\varphi - \frac{5}{27}}} \right) = \sqrt[3]{\frac{8}{27}} = \frac{2}{3} \quad \text{so,} \quad \xi^3 = 2\xi + \varphi \quad (Eq.7)$$

Equation 8

Noting that $D = \sqrt{3} \sin \alpha - \cos \alpha$ we find :

$$\begin{aligned}
 & (9\varphi^3 + \varphi + 6\xi - 3\varphi^3\xi^2)^2 - (\varphi + 3\xi)^2 (1 - 3\varphi^4(3\sqrt{5} - 2\varphi^3\xi + \varphi^2\xi^2)) = 0 \quad (\text{expand and substitute } \xi^3 \text{ with Eq.7}) \\
 & = (9\varphi^3 + \varphi)^2 + 36\xi^2 + 9\varphi^6\xi(2\xi + \varphi) + 2(9\varphi^3 + \varphi)(6\xi - 3\varphi^3\xi^2) - 36\varphi^3(2\xi + \varphi) - \varphi^2 (1 - 3\varphi^4(3\sqrt{5} - 2\varphi^3\xi + \varphi^2\xi^2)) \\
 & \quad - 6\varphi (\xi - 3\varphi^4 (3\sqrt{5}\xi - 2\varphi^3\xi^2 + \varphi^2(2\xi + \varphi))) - 9 (\xi^2 - 3\varphi^4 (3\sqrt{5}\xi^2 - 2\varphi^3(2\xi + \varphi) + \varphi^2\xi(2\xi + \varphi))) = 0 \\
 & = (81\varphi^6 + 18\varphi^4 + \varphi^2 - 36\varphi^4 - \varphi^2 + 9\varphi^6\sqrt{5} + 18\varphi^8 - 54\varphi^8) \quad (\text{all 3 orders of } \xi \text{ sum to zero}) \\
 & \quad + (9\varphi^7 + 12(9\varphi^3 + \varphi) - 72\varphi^3 + 6\varphi^9 - 6\varphi + 54\varphi^5\sqrt{5} + 36\varphi^7 - 108\varphi^8 + 27\varphi^7)\xi \\
 & \quad + (36 + 18\varphi^6 - 6\varphi^3(9\varphi^3 + \varphi) + 3\varphi^8 - 36\varphi^8 - 9 + 81\varphi^4\sqrt{5} + 54\varphi^6)\xi^2 = 0
 \end{aligned}$$

$$\sqrt{1 - 3\varphi^4(3\sqrt{5} - 2\varphi^3\xi + \varphi^2\xi^2)} = \frac{9\varphi^3 + \varphi + 6\xi - 3\varphi^3\xi^2}{\varphi + 3\xi} \quad (\text{Eq.8})$$

Solve for cos of alpha

Substitute Eq.6 and Eq.8 into Eq.3

$$\begin{aligned}
 \gamma &= \frac{-3 + \sqrt{9 - 12\Gamma \cos \alpha (\Gamma \cos \alpha - 3)}}{2\Gamma \cos \alpha} = \frac{-1 + \sqrt{1 - 3\varphi^4(3\sqrt{5} - 2\varphi^3\xi + \varphi^2\xi^2)}}{3\varphi^2 - \varphi^3\xi} = \frac{-1 + \frac{9\varphi^3 + \varphi + 6\xi - 3\varphi^3\xi^2}{\varphi + 3\xi}}{3\varphi^2 - \varphi^3\xi} \\
 \gamma &= \frac{-\varphi - 3\xi + 9\varphi^3 + \varphi + 3\xi + 3(3\varphi^2 - \varphi^4)\xi - 3\varphi^3\xi^2}{(\varphi + 3\xi)(3\varphi^2 - \varphi^3\xi)} = \frac{3\varphi + 3\xi}{\varphi + 3\xi} \\
 \cos \alpha &= \frac{1}{\sqrt{1 + \tan^2 \alpha}} = \frac{1}{\sqrt{1 + \frac{\gamma^2}{3}}} = \frac{1}{\sqrt{1 + \frac{1}{3} \left(\frac{3\varphi + 3\xi}{\varphi + 3\xi} \right)^2}} = \frac{\varphi + 3\xi}{2\sqrt{\varphi^2 + 3\xi(\varphi + \xi)}}
 \end{aligned}$$

Equation 9

$$\begin{aligned}
 \Gamma &= \cos \alpha [3 - \gamma] = \left(\frac{\varphi + 3\xi}{2\sqrt{\varphi^2 + 3\xi(\varphi + \xi)}} \right) \left[3 - \frac{3\varphi + 3\xi}{\varphi + 3\xi} \right] \\
 \Gamma &= \frac{3\xi}{\sqrt{\varphi^2 + 3\xi(\varphi + \xi)}} \quad (\text{Eq.9})
 \end{aligned}$$

Solve for D

Substitute Eq.6 and Eq.9 into Eq.5

$$D = \frac{\varphi^4 \Gamma}{9\varphi^2 - 2\Gamma \cos \alpha} = \frac{\varphi \Gamma}{3\xi} = \frac{\varphi}{\sqrt{\varphi^2 + 3\xi(\varphi + \xi)}}$$

Volume for Snub Dodecahedron (Thirteenth Archimedean Solid)

$$\text{Icosa symmetry: } \cos \beta = \frac{\varphi}{\sqrt{3}} \quad \sin \beta = \frac{1}{\varphi\sqrt{3}} \quad \overline{ab} = \frac{1}{2\sqrt{3}} \quad \overline{ac} = \frac{\overline{ab}}{\tan \beta} = \frac{\varphi^2}{2\sqrt{3}}$$

$$\text{Radius to triangle face: } r_{\text{triangle}} = \frac{\overline{ac}}{D} = \frac{\varphi}{2\sqrt{3}} \sqrt{\varphi^2 + 3\xi(\varphi + \xi)}$$

Circumradius (radius to vertex):

$$r_{\text{circumradius}} = \sqrt{r_{\text{triangle}}^2 + (2 \sin \frac{\pi}{3})^{-2}} = \sqrt{\frac{\varphi^2 (\varphi^2 + 3\xi(\varphi + \xi)) + 4}{12}} = \frac{1}{2} \sqrt{\frac{\varphi^4 + 4 + 3\varphi^2 \xi(\varphi + \xi)}{3}}$$

Inradius (radius to pentagon face):

$$r_{\text{pentagon}} = \sqrt{r_{\text{circumradius}}^2 - (2 \sin \frac{\pi}{5})^{-2}} = \sqrt{\frac{\varphi^4 + 4 + 3\varphi^2 \xi(\varphi + \xi)}{12} - \frac{\varphi}{\sqrt{5}}} = \frac{\varphi}{2} \sqrt{\frac{1}{\varphi\sqrt{5}} + \xi(\varphi + \xi)}$$

Midradius (radius to edge bisector):

$$r_{\text{midradius}} = \sqrt{r_{\text{triangle}}^2 + (2 \tan \frac{\pi}{3})^{-2}} = \sqrt{\frac{\varphi^2 (\varphi^2 + 3\xi(\varphi + \xi)) + 1}{12}} = \frac{1}{2} \sqrt{\frac{\varphi^4 + 1 + 3\varphi^2 \xi(\varphi + \xi)}{3}}$$

$$\begin{aligned} \text{Volume}_{\text{SnubDodecahedron}} &= N_{\text{triangle}} \times \text{Area}_{\text{triangle}} \times \frac{1}{3} \times r_{\text{triangle}} + N_{\text{pentagon}} \times \text{Area}_{\text{pentagon}} \times \frac{1}{3} \times r_{\text{pentagon}} \\ &= 80 \frac{\sqrt{3}}{4} \frac{1}{3} \frac{\varphi}{2\sqrt{3}} \sqrt{\varphi^2 + 3\xi(\varphi + \xi)} + 12 \frac{5}{4} \sqrt{\frac{\varphi^3}{\sqrt{5}}} \frac{1}{3} \frac{\varphi}{2} \sqrt{\frac{1}{\varphi\sqrt{5}} + \xi(\varphi + \xi)} \end{aligned}$$

$$\text{Volume}_{\text{SnubDodecahedron}} = \frac{10\varphi}{3} \sqrt{\varphi^2 + 3\xi(\varphi + \xi)} + \frac{\varphi^2}{2} \sqrt{5 + 5\sqrt{5}\varphi\xi(\varphi + \xi)}$$

Part 3. Calculations of the Snub Dodecahedron

Part 3. is a Python script that calculates the volume of the Snub Dodecahedron using the five methods described above:

- Method 1 by Harold Scott MacDonald Coxeter as shown in Wikipedia
- Method 2 by Harold Scott MacDonald Coxeter as shown in MathWorld
- Method 3 by Mark Shelby Adams
- Method 4 by Harish Chandra Rajpoot
- Method 5 by 3D Numerical

```

In [2]: # Python 3.7 calculations to the Snub Dodecahedron

import mpmath as mp

#####
class Calculations_of_Snub_Dodecahedron_Coxeter_Wikipedia(object):
    '''Volume calculation of Snub Dodecahedron from
    H.S.M. Coxeter's work as shown in Wikipedia
    https://en.wikipedia.org/wiki/Snub_dodecahedron'''

    def __init__(self):
        mp.mp.dps = 55
        self.run_calculation()

    def run_calculation(self):
        self.digits = 53
        self.phi = (mp.sqrt(5) + 1) / 2 # Golden Section
        self.xi = ( self.phi / 2 + (mp.mpf(1)/2) * mp.sqrt( self.phi - mp.mpf(5)/27 ))** ((mp.mpf(1)/3) )
+ (
        ( self.phi / 2 - (mp.mpf(1)/2) * mp.sqrt( self.phi - mp.mpf(5)/27 ))** ((mp.mpf(1)/3) )
)

        part1 = ( 12 * self.xi**2) * ( 3 * self.phi + 1)
        part2 =          self.xi          * ( 36 * self.phi + 7)
        part3 =          ( 53 * self.phi + 6)
        part4 = 4 * ( part1 - part2 - part3)
        part5 = 3 * ((2 * mp.sqrt( 3 - self.xi * self.xi)) ** 3)
        self.volume_coxeter_wikipedia = part4 / part5
        print("")
        print("Volume calculation of Snub Dodecahedron from")
        print(" H.S.M. Coxeter's work as shown in Wikipedia")
        print(" https://en.wikipedia.org/wiki/Snub_dodecahedron")
        print('Coxeter Wikipedia Volume = %s'%(mp.nstr(self.volume_coxeter_wikipedia, self.digits)))

    def print_volume(self):
        print('Coxeter Wikipedia Volume = %s'%(mp.nstr(self.volume_coxeter_wikipedia, self.digits)))
        return

#####
class Calculations_of_Snub_Dodecahedron_Coxeter_Mathworld(object):
    '''Volume calculation of Snub Dodecahedron from
    H.S.M. Coxeter's work as shown in Eric Weisstein's MathWorld
    http://mathworld.wolfram.com/SnubDodecahedron.html'''

    def __init__(self):
        self.run_calculation()

    def finder_f(self, f_value):
        self.f_value = f_value
        self.f_counter += 1
        self.delta = (
            mp.mpf(187445810737515625) -
            mp.mpf(182124351550575000) * self.f_value**2 +
            mp.mpf( 6152923794150000) * self.f_value**4 +
            mp.mpf( 1030526618040000) * self.f_value**6 +
            mp.mpf( 1622231919360000) * self.f_value**8 -
            mp.mpf( 3195335070720) * self.f_value**10 +
            mp.mpf( 2176782336) * self.f_value**12)
        return(self.delta)

    def run_calculation(self):
        mp.mp.dps = 60
        self.digits = 53
        tolerance = mp.mpf('1.0e-55')
        self.f_counter = 0
        mp.findroot(self.finder_f, mp.mpf(36.0), tol=tolerance,
            solver='halley', maxsteps=2000, verbose=False)
        self.volume_coxeter_mathworld = self.f_value
        print("")
        print("Volume calculation of Snub Dodecahedron (%d iterations) from"%(self.f_counter))
        print(" H.S.M. Coxeter's work as shown in Eric Weisstein's MathWorld")
        print(" http://mathworld.wolfram.com/SnubDodecahedron.html")
        print('Coxeter Mathworld Volume = %s'%(mp.nstr(self.volume_coxeter_mathworld, self.digits)))
        return

    def print_volume(self):
        print('Coxeter Mathworld Volume = %s'%(mp.nstr(self.volume_coxeter_mathworld, self.digits)))
        return

#####

```



```

class Calculations_of_Snub_Dodecahedron_Adams(object):
    '''Volume calculation of the Snub Dodecahedron as shown from
    Mark Adams's book Archimedean & Platonic Solids
    https://zenodo.org/record/2563268#.XGteD7pKhhE'''

    def __init__(self):
        mp.mp.dps = 55
        self.run_calculations()

    def run_calculations(self):
        self.digits = 52
        self.phi = (mp.sqrt(5) + 1) / 2 # Golden Section
        self.xi = ( self.phi / 2 + (mp.mpf(1)/2) * mp.sqrt( self.phi - mp.mpf(5)/27 ))** ((mp.mpf(1)/3) )
+ (
        ( self.phi / 2 - (mp.mpf(1)/2) * mp.sqrt( self.phi - mp.mpf(5)/27 ))** ((mp.mpf(1)/3) )
)
        self.radius_triangle = (self.phi / (2 * mp.sqrt(3))) * mp.sqrt( self.phi**2 + 3* self.xi * (self.ph
i + self.xi))
        self.radius_circumradius = mp.sqrt( (self.phi**2 * ( self.phi**2 + 3* self.xi * (self.phi + self.xi
)) + 4) / 12)
        self.radius_pentagon = (self.phi/2) * mp.sqrt( (1 / (self.phi*mp.sqrt(5))) + self.xi*(self.phi + se
lf.xi))
        self.radius_mid = (mp.mpf(1)/2) * mp.sqrt(( self.phi**4 + 1 + 3 * self.phi**2 * self.xi * (self.phi
+ self.xi)) / 3)
        self.volume_adams = (10*self.phi/3)*mp.sqrt( self.phi**2 + 3* self.xi * (self.phi + self.xi)) + (
        (self.phi**2/2)*mp.sqrt( 5 + 5 * mp.sqrt(5) * self.phi * self.xi * (self.phi + self.xi)))
        print("")
        print("Volume calculation of the Snub Dodecahedron as shown from")
        print(" Mark Adams's book Archimedean & Platonic Solids")
        print(" https://zenodo.org/record/2563268#.XGteD7pKhhE")
        print('Xi = %s'%(mp.nstr(self.xi, self.digits)))
        print('Radius_triangle = %s'%(mp.nstr(self.radius_triangle, self.digits)))
        print('Radius_circumradius = %s'%(mp.nstr(self.radius_circumradius, self.digits)))
        print('Radius_pentagon = %s'%(mp.nstr(self.radius_pentagon, self.digits)))
        print('Radius_mid = %s'%(mp.nstr(self.radius_mid, self.digits)))
        print('Adams Volume = %s'%(mp.nstr(self.volume_adams, self.digits+1)))

    def print_volume(self):
        print('Adams Volume = %s'%(mp.nstr(self.volume_adams, self.digits+1)))
        return

#####
class Calculations_of_Snub_Dodecahedron_Rajpoot(object):
    '''Volume calculation of the Snub Dodecahedron from Harish Chandra Rajpoot's paper
    https://works.bepress.com/harishchandraraipoot_hcraipoot/27/'''

    def __init__(self):
        mp.mp.dps = 55
        self.run_calculations()

    def r_function(self, x):
        value = 256*(mp.mpf( 3) - mp.sqrt(5)) * x**8 - (
        128*(mp.mpf(13) - 2 * mp.sqrt(5)) * x**6 ) + (
        32*(mp.mpf(35) - 3 * mp.sqrt(5)) * x**4 ) - (
        16*(mp.mpf(19) - mp.sqrt(5)) * x**2 ) + (
        (mp.mpf(29) - mp.sqrt(5)) )
        return(value)

    def r_function_prime(self, x):
        value = 2048*(mp.mpf( 3) - mp.sqrt(5)) * x**7 - (
        768*(mp.mpf(13) - 2 * mp.sqrt(5)) * x**5 ) + (
        128*(mp.mpf(35) - 3 * mp.sqrt(5)) * x**3 ) - (
        32*(mp.mpf(19) - mp.sqrt(5)) * x )
        return(value)

    def run_calculations(self):
        self.digits = 52
        self.iterations = 7
        self.C = mp.mpf(2.3) # Starting value
        for i in range(self.iterations):
            i=i
            self.C = self.C - self.r_function(self.C) / self.r_function_prime(self.C)
            self.volume_rajpoot = 20 * mp.sqrt(3 * self.C**2 - 1) / mp.mpf(3) + (
            mp.sqrt(( 10 * (5 + 2 * mp.sqrt(5)) * self.C**2 - 5 * (7 + 3*mp.sqrt(5)))/mp.mpf(2)))
        print("")
        print("Volume calculation of the Snub Dodecahedron from Harish Chandra Rajpoot's paper")
        print("https://works.bepress.com/harishchandraraipoot_hcraipoot/27/")
        print('Circumradius (%d iterat) = %s'%(self.iterations, mp.nstr(self.C, self.digits)))
        print('Rajpoot Volume = %s'%(mp.nstr(self.volume_rajpoot, self.digits+1)))

    def print_volume(self):

```

```

print('Rajpoot Volume          = %s'%(mp.nstr(self.volume_rajpoot, self.digits+1)))
return

#####
class Calculations_of_Snub_Dodecahedron_3D_Numerical(object):
    '''Two triangle objects are define as adjacent triangles on a regular icosahedron.
    A root finder algorithm is applied to one point on the plane of each triangle so
    that the distance of a side of an inscribed snub triangle is equal to the side
    of a non inscribed snub triangle.'''

    def __init__(self):
        mp.mp.dps = 55
        self.verbose= True
        self.snub_dodecahedron_numerical_findroot()

    class Point(object):
        def __init__(self, parent, x, y, z):
            self.parent= parent
            self.x= x
            self.y= y
            self.z= z

        def make_copy(self):
            self.mcopy= self.parent.Point(self.parent,self.x,self.y,self.z)
            return self.mcopy

        def make_vector_to(self, endpoint):
            self.vector = self.parent.Point(self.parent,
                endpoint.x-self.x,
                endpoint.y-self.y,
                endpoint.z-self.z)
            return self.vector

        def distance_to(self, endpoint):
            self.distance = mp.sqrt(
                (endpoint.x-self.x)**2 +
                (endpoint.y-self.y)**2 +
                (endpoint.z-self.z)**2 )
            return self.distance

        def add_vector(self, point):
            self.x += point[0]
            self.y += point[1]
            self.z += point[2]
            return

        def scale(self, ratio):
            self.ratio = ratio
            self.scaled_xyz = [
                self.x * self.ratio,
                self.y * self.ratio,
                self.z * self.ratio]
            return self.scaled_xyz

    class Triangle(object):
        def __init__(self, parent, pointA, pointB, pointC):
            self.parent= parent
            self.point_A= parent.Point(self.parent, pointA[0], pointA[1], pointA[2])
            self.point_B= parent.Point(self.parent, pointB[0], pointB[1], pointB[2])
            self.point_C= parent.Point(self.parent, pointC[0], pointC[1], pointC[2])
            self.center = parent.Point(self.parent,
                (self.point_A.x + self.point_B.x + self.point_C.x) /3,
                (self.point_A.y + self.point_B.y + self.point_C.y) /3,
                (self.point_A.z + self.point_B.z + self.point_C.z) /3)
            self.ratio_1= mp.mp.mpf('.1')
            self.ratio_2= mp.mp.mpf('.1')
            self.vector_to_g_1 = self.center.make_vector_to( self.point_B)
            self.vector_to_g_2 = self.point_B.make_vector_to(self.point_C)
            self.vector_to_j_1 = self.center.make_vector_to( self.point_C)
            self.vector_to_j_2 = self.point_C.make_vector_to(self.point_A)

        def set_ratio_1(self, ratio_1):
            self.ratio_1= ratio_1
            return

        def set_ratio_2(self, ratio_2):
            self.ratio_2= ratio_2
            return

        def calculate(self):
            self.point_g = self.center.make_copy()

```

```

self.point_g.add_vector(self.vector_to_g_1.scale(self.ratio_1))
self.point_g.add_vector(self.vector_to_g_2.scale(self.ratio_2))
self.point_j = self.center.make_copy()
self.point_j.add_vector(self.vector_to_j_1.scale(self.ratio_1))
self.point_j.add_vector(self.vector_to_j_2.scale(self.ratio_2))
# Ratio from center of eq. triangle to vertex by edge length is sqrt(3)
self.distance_g_j = self.point_g.distance_to(self.center) * mp.sqrt(3)
return

def info(self):
    digits = 5
    D1 = mp.nstr(self.point_A.distance_to(self.point_B), digits)
    D2 = mp.nstr(self.point_B.distance_to(self.point_C), digits)
    D3 = mp.nstr(self.point_C.distance_to(self.point_A), digits)
    self.parent.log.info('Base: %s\n      %s\n      %s'%(D1,D2,D3))
    return

def snub_dodecahedron_numerical_findroot(self):
    self.make_icosahedron()
    self.triangle_1 = self.Triangle(self, self.icosahedron[1], self.icosahedron[0], self.icosahedron[2])
    self.triangle_2 = self.Triangle(self, self.icosahedron[3], self.icosahedron[2], self.icosahedron[0])
    self.run_numerical_solution()
    self.digits = 52
    self.D = self.triangle_1.distance_g_j
    self.phi = (mp.sqrt(5) + 1) / 2 # Golden Section
    self.radius_triangle_unit_icosahedron = self.phi**2/(2*mp.sqrt(3))
    self.r_tri = self.radius_triangle_unit_icosahedron / self.D
    self.r_circ = mp.sqrt( self.r_tri**2 + (2*mp.sin(mp.pi/3))**(-2) )
    self.r_pent = mp.sqrt( self.r_circ**2 - (mp.mpf(2)*mp.sin((mp.pi/5)))**(-2))
    self.r_mid = mp.sqrt( self.r_tri**2 + (2*mp.tan(mp.pi/3))**(-2) )
    self.volume_numerical = 80 * mp.sqrt(3)/4 * (mp.mpf(1)/3) * self.r_tri + (
        12 * (mp.mpf(5)/4) * mp.sqrt( self.phi**3 / mp.sqrt(5)) * (mp.mpf(1)/3) * self.r_pent)
    print("")
    print("Volume calculation of Snub Dodecahedron from %d 3D numerical iterations"%(
        self.finder_counter))
    print('Radius_triangle          = %s'%(mp.nstr(self.r_tri , self.digits)))
    print('Radius_circumradius        = %s'%(mp.nstr(self.r_circ, self.digits)))
    print('Radius_pentagon              = %s'%(mp.nstr(self.r_pent, self.digits)))
    print('Radius_mid                   = %s'%(mp.nstr(self.r_mid , self.digits)))
    print('Numerical Volume             = %s'%( mp.nstr(self.volume_numerical, self.digits+1)))

def print_volume(self):
    print('Numerical Volume          = %s'%( mp.nstr(self.volume_numerical, self.digits+1)))
    return

def run_numerical_solution(self):
    mp.mp.dps = 60
    self.verbose = True
    self.finder_counter = 0
    tolerance = mp.mpf('1.0e-55')
    mp.findroot(self.finder_f, mp.mpc('0.1','0.1'), tol=tolerance,
        solver='halley', maxsteps=2000, verbose=False)
    return

def finder_f(self, ratio):
    'Called by mp.findroot'
    self.finder_counter += 1
    self.triangle_1.set_ratio_1(ratio.real)
    self.triangle_2.set_ratio_1(ratio.real)
    self.triangle_1.set_ratio_2(ratio.imag)
    self.triangle_2.set_ratio_2(ratio.imag)
    self.triangle_1.calculate()
    self.triangle_2.calculate()
    self.distance_j_gprime = self.triangle_1.point_j.distance_to(self.triangle_2.point_g)
    self.distance_g_gprime = self.triangle_1.point_g.distance_to(self.triangle_2.point_g)
    self.delta_1 = self.triangle_1.distance_g_j - self.distance_j_gprime
    self.delta_2 = self.triangle_1.distance_g_j - self.distance_g_gprime
    self.delta_distance = mp.mpc(self.delta_1, self.delta_2)
    return(self.delta_distance)

def make_icosah(self, verbose=0, unit_edge_length=1 ):
    self.icosah_faces = [
        ( 0, 1, 2) ,( 0, 2, 3) ,( 0, 3, 4) ,( 0, 4, 5) ,( 0, 5, 1),
        (11, 6, 7) ,(11, 7, 8) ,(11, 8, 9) ,(11, 9, 10) ,(11, 10, 6),
        ( 1, 2, 6) ,( 2, 3, 7) ,( 3, 4, 8) ,( 4, 5, 9) ,( 5, 1, 10),
        ( 6, 7, 2) ,( 7, 8, 3) ,( 8, 9, 4) ,( 9, 10, 5) ,(10, 6, 1)]
    c63 = 1 / mp.sqrt(5) # Cos(a) a = arctan(2) = 63.434... degrees
    s63 = 2 / mp.sqrt(5) # Sin(a) a = arctan(2) = 63.434... degrees
    c72 = mp.sqrt((3-mp.sqrt(5))/8) # Cos(72)
    s72 = mp.sqrt((5+mp.sqrt(5))/8) # Sin(72)
    c36 = mp.sqrt((3+mp.sqrt(5))/8) # Cos(36)
    s36 = mp.sqrt((5-mp.sqrt(5))/8) # Sin(36)

```

```

self.icoso = [
    [ 0, 0, 1],
    [ s63, 0, c63],
    [ s63*c72, s63*s72, c63],
    [-s63*c36, s63*s36, c63],
    [-s63*c36, -s63*s36, c63],
    [ s63*c72, -s63*s72, c63],
    [ s63*c36, s63*s36, -c63],
    [-s63*c72, s63*s72, -c63],
    [-s63, 0, -c63],
    [-s63*c72, -s63*s72, -c63],
    [ s63*c36, -s63*s36, -c63],
    [ 0, 0, -1]]
if unit_edge_length:
    self.point_A= self.Point(self, self.icoso[0][0],self.icoso[0][1],self.icoso[0][2])
    self.point_B= self.Point(self, self.icoso[1][0],self.icoso[1][1],self.icoso[1][2])
    adjust = 1 / self.point_A.distance_to(self.point_B)
    for i, points in enumerate(self.icoso):
        for j, xyz in enumerate(points):
            self.icoso[i][j] *= adjust
if verbose:
    digits = 3
    for i, xyz in enumerate(self.icoso):
        x = mp.nstr(xyz[0], digits)
        y = mp.nstr(xyz[1], digits)
        z = mp.nstr(xyz[2], digits)
        print('V %d (%s, %s %s)'%(i+1, x,y,z))
return

```

```

calculation_coxeter_w = Calculations_of_Snub_Dodecahedron_Coxeter_Wikipedia()
calculation_coxeter_m = Calculations_of_Snub_Dodecahedron_Coxeter_Mathworld()
calculation_adams = Calculations_of_Snub_Dodecahedron_Adams()
calculation_rajpoot = Calculations_of_Snub_Dodecahedron_Rajpoot()
calculation_numerical = Calculations_of_Snub_Dodecahedron_3D_Numerical()

```

```

print("")
print("Volumes:")
calculation_coxeter_w.print_volume()
calculation_coxeter_m.print_volume()
calculation_adams.print_volume()
calculation_rajpoot.print_volume()
calculation_numerical.print_volume()

```

Volume calculation of Snub Dodecahedron from
 H.S.M. Coxeter's work as shown in Wikipedia
https://en.wikipedia.org/wiki/Snub_dodecahedron
 Coxeter Wikipedia Volume = 37.616649962733362975777673671302714340355289873488099

Volume calculation of Snub Dodecahedron (44 iterations) from
 H.S.M. Coxeter's work as shown in Eric Weisstein's MathWorld
<http://mathworld.wolfram.com/SnubDodecahedron.html>
 Coxeter Mathworld Volume = 37.616649962733362975777673671302714340355289873488099

Volume calculation of the Snub Dodecahedron as shown from
 Mark Adams's book Archimedean & Platonic Solids
<https://zenodo.org/record/2563268#.XGteD7pKhE>
 Xi = 1.715561499697367834681278888889983371197187471404009
 Radius_triangle = 2.077089659743208599411307935302249276745292242657295
 Radius_circumradius = 2.155837375115639701836629076693058277016851218774812
 Radius_pentagon = 1.980915947281840739000205339530447996567952552681664
 Radius_mid = 2.097053835252087992403959052348286240030839730581031
 Adams Volume = 37.616649962733362975777673671302714340355289873488099

Volume calculation of the Snub Dodecahedron from Harish Chandra Rajpoot's paper
https://works.bepress.com/harishchandarajpoot_hcrajpoot/27/
 Circumradius (7 iterat) = 2.155837375115639701836629076693058277016851218774812
 Rajpoot Volume = 37.616649962733362975777673671302714340355289873488099

Volume calculation of Snub Dodecahedron from 800 3D numerical iterations
 Radius_triangle = 2.077089659743208599411307935302249276745292242657295
 Radius_circumradius = 2.155837375115639701836629076693058277016851218774812
 Radius_pentagon = 1.980915947281840739000205339530447996567952552681664
 Radius_mid = 2.097053835252087992403959052348286240030839730581031
 Numerical Volume = 37.616649962733362975777673671302714340355289873488099

Volumes:
 Coxeter Wikipedia Volume = 37.616649962733362975777673671302714340355289873488099
 Coxeter Mathworld Volume = 37.616649962733362975777673671302714340355289873488099
 Adams Volume = 37.616649962733362975777673671302714340355289873488099
 Rajpoot Volume = 37.616649962733362975777673671302714340355289873488099
 Numerical Volume = 37.616649962733362975777673671302714340355289873488099

References

- Pacioli, Luca. "Divina proportione" 1509.
 - <https://archive.org/details/divinaproportion00paci/page/n27> (<https://archive.org/details/divinaproportion00paci/page/n27>)
- Kepler, Johannes "Harmonices Mundi." 1619.
 - <https://archive.org/details/ioanniskepplerih00kepl/page/n85> (<https://archive.org/details/ioanniskepplerih00kepl/page/n85>)
- Coxeter, H. S. M.; Longuet-Higgins, M. S.; and Miller, J. C. P. "Uniform Polyhedra."
 - Phil. Trans. Roy. Soc. London Ser. A 246, 401-450, 1954.
 - <https://royalsocietypublishing.org/doi/abs/10.1098/rsta.1954.0003> (<https://royalsocietypublishing.org/doi/abs/10.1098/rsta.1954.0003>).
- Weisstein, Eric W. "Snub Dodecahedron." From MathWorld--A Wolfram Web Resource.
 - <http://mathworld.wolfram.com/SnubDodecahedron.html> (<http://mathworld.wolfram.com/SnubDodecahedron.html>)
- Adams, Mark Shelby. "Archimedean & Platonic Solids." January 7, 1985.
 - <https://doi.org/10.5281/zenodo.2563268> (<https://doi.org/10.5281/zenodo.2563268>).
- Harish Chandra Rajpoot, H. C. R. "Optimum Solution of Snub Dodecahedron
 (an Archimedean Solid) by Using 'HCR's Theory of Polygon' & 'Newton-Raphson Method,'" 2015.
 - https://works.bepress.com/harishchandarajpoot_hcrajpoot/27/ (https://works.bepress.com/harishchandarajpoot_hcrajpoot/27/).
- "Snub Dodecahedron." In Wikipedia
 - https://en.wikipedia.org/w/index.php?title=Snub_dodecahedron (https://en.wikipedia.org/w/index.php?title=Snub_dodecahedron).
- This essay in jupyter notebook format:
 - https://nbviewer.jupyter.org/url/web.archive.org/web/20190308012045/http%3A//geod.com/Snub_Dodecahedron.ipynb
https://nbviewer.jupyter.org/url/web.archive.org/web/20190308012045/http%3A//geod.com/Snub_Dodecahedron.ipynb

End of essay, Imaginative solutions to the Snub Dodecahedron