icosahedron_calcs

March 22, 2024

0.0.1 Icosahedron construction (20 Faces)

Working with sperical coordinates

$$\begin{pmatrix} x \\ y \\ z \end{pmatrix} = \begin{pmatrix} r * sin(\theta) * cos(\phi) \\ r * sin(\theta) * sin(\phi) \\ r * cos(\theta) \end{pmatrix} \tag{1}$$

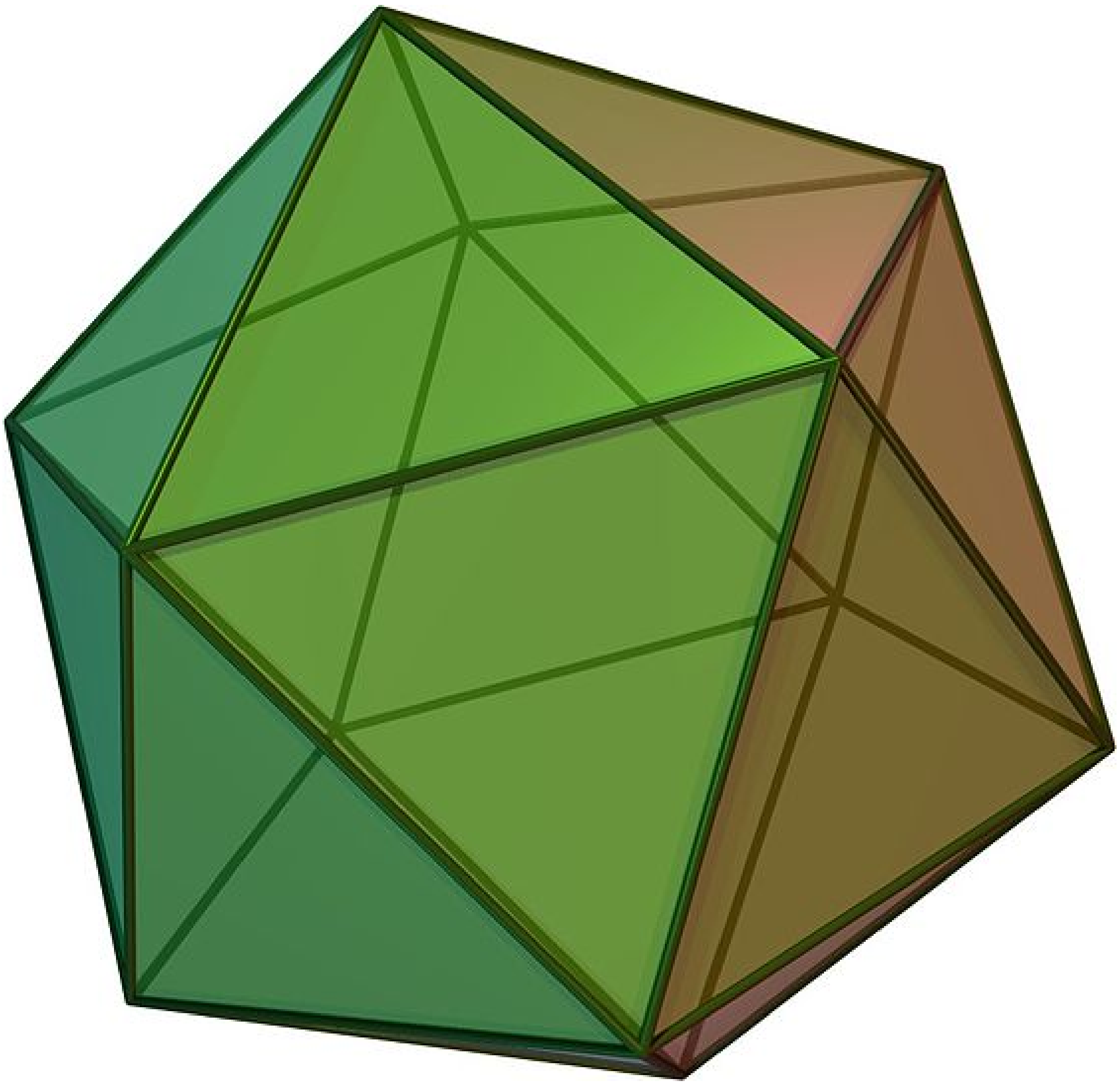
where

$$\theta = [0, \pi[\phi = [0, 2\pi[\tag{2}$$

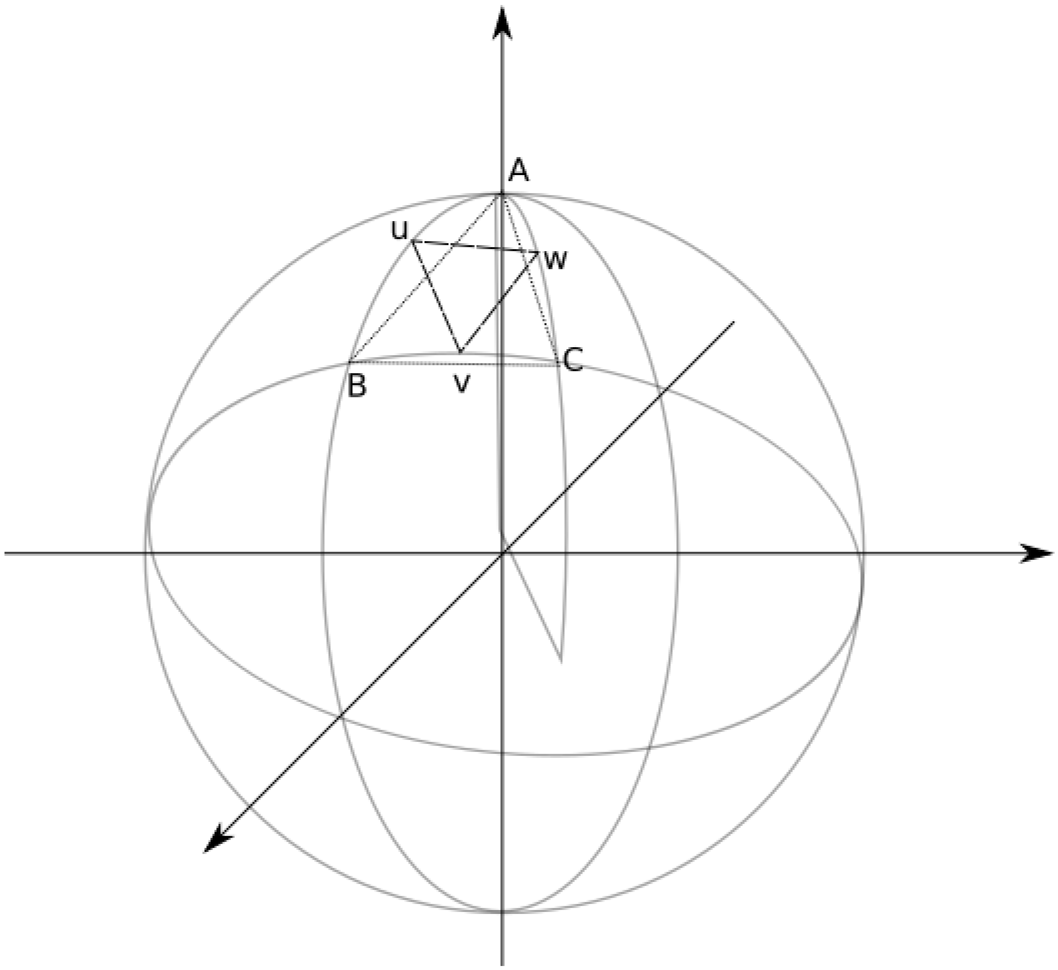
Cosider a equilateral triangle of the base 20-faces icosahedron. The spanning angle of this triangle along the sphere touching the vertices of the icosahedron is

$$\alpha = \frac{2\pi}{5}$$

since a set of five face sharing a common vertex form a pentagram.



For calculating the equilaterial and the isoscele triangle side length we consider three points A, B and C forming a vertex of the base icosahedron.



These points lie at $A : (r, 0, 0)$, $B : (r, \Theta_0, 0)$, $C : (r, \Theta_0, \alpha)$ And the points u,v,w at the center of the sides at $u = (r, \Theta_0/2, 0)$, $v = (r, \Theta_0 * p, \alpha/2)$ and $w = (r, \Theta_0/2, \alpha)$ with Θ_0 being

$$\Theta_0 = 2asin(\frac{a}{2r})$$

The side length of the equilateral triangle is simply $\overline{uv}$ and the side length of the isoscele triangle is $\overline{Au}$ although it would be probably possible to determine the factor p analytically for the sake of simplicity it is solved numerically by minimizing the differences between $\overline{uv}$, $\overline{uw}$ and $\overline{vw}$

0.1 Definitions

[2]:
```
a=4#side length of the base 20-face icosahedron
h=2.5 # height of the rhomboid extending the icosphere
```

0.2 Calculations

[3]:
```
import math

def sph_cart(r,theta,phi):
    return [r*math.sin(theta)*math.cos(phi), r*math.sin(theta)*math.sin(phi),␣
 ↪r*math.cos(theta)]
```

```
def sph_cart_array(p):
    return sph_cart(p[0], p[1], p[2])

def cart_dist(p1,p2):
    dx = p2[0] - p1[0]
    dy = p2[1] - p1[1]
    dz = p2[2] - p1[2]
    return math.sqrt(dx*dx + dy*dy + dz*dz)
```

[4]:
```
alpha = 2.*math.pi/5.
r = a * math.sin(alpha)

theta0 = 2.*math.asin(a/(2.*r))
p=1.
d_p =-0.01
delta_f = 0.00000001
A = [r, 0, 0]
B = [r, theta0, 0]
C = [r, theta0, alpha]
u = [r, theta0*0.5, 0]
v = [r, theta0*p, alpha/2.]
w = [r, theta0*0.5, alpha]

d_uw = cart_dist(sph_cart_array(u),sph_cart_array(w))
d_uv = cart_dist(sph_cart_array(u),sph_cart_array(v))
d_wv = cart_dist(sph_cart_array(w),sph_cart_array(v))
dev_d = max(d_uw,d_uv,d_wv) - min(d_uw,d_uv,d_wv)
n_it = 0
p_new = p + d_p
while dev_d > delta_f and n_it < 200:

    u = [r, theta0*0.5, 0]
    v = [r, theta0*p_new, alpha/2.]
    w = [r, theta0*0.5, alpha]

    d_uw = cart_dist(sph_cart_array(u),sph_cart_array(w))
    d_uv = cart_dist(sph_cart_array(u),sph_cart_array(v))
    d_wv = cart_dist(sph_cart_array(w),sph_cart_array(v))
    dev_d_new = max(d_uw,d_uv,d_wv) - min(d_uw,d_uv,d_wv)
    #print("p_new: {}, dev_d_new: {}".format(p_new,dev_d_new))
    if dev_d_new < dev_d:
        p = p_new
        dev_d = dev_d_new
        #print("dev_d: {}".format(dev_d))
    else:
        # p = p-d_p
```

```
        d_p = -d_p*0.75
        #print("d_p: {}".format(d_p))
    p_new = p + d_p
    n_it += 1

    if dev_d <= delta_f:
        print("finished!!, n_it: {}, p is {}".format(n_it, p))
```

```
finished!!, n_it: 75, p is 0.9187762667181671
```

0.3 Consistency Checks

the distances $\overline{uv}$, $\overline{uw}$ and $\overline{vw}$ should be equal, also the distances $\overline{Au}$,$\overline{Aw}$, $\overline{Bu}$, $\overline{Bv}$, $\overline{Cv}$ and $\overline{Cw}$ should be equal

[5]:
```
cart_dist(sph_cart_array(u),sph_cart_array(w))
```

[5]: 2.351141009169892

[6]:
```
cart_dist(sph_cart_array(u),sph_cart_array(v))
```

[6]: 2.3511410022106607

[7]:
```
cart_dist(sph_cart_array(w),sph_cart_array(v))
```

[7]: 2.3511410022106607

[8]:
```
cart_dist(sph_cart_array(A),sph_cart_array(w))
```

[8]: 2.0791353040620346

[9]:
```
cart_dist(sph_cart_array(A),sph_cart_array(u))
```

[9]: 2.079135304062035

[10]:
```
cart_dist(sph_cart_array(B),sph_cart_array(u))
```

[10]: 2.079135304062035

[11]:
```
cart_dist(sph_cart_array(B),sph_cart_array(v))
```

[11]: 2.079135304062035

[12]:
```
cart_dist(sph_cart_array(C),sph_cart_array(v))
```

[12]: 2.079135304062035

[13]:
```
cart_dist(sph_cart_array(C),sph_cart_array(w))
```

[13]: 2.079135304062035

[14]:
```
def calc_rhomboid(h,r,d):
    h_base = math.sqrt(r*r - d*d/4.)
    return d + 2.*((d/2.)/h_base)*h
# rhomboid calculations

d_equidistant = cart_dist(sph_cart_array(u),sph_cart_array(w))
d_isoscele = cart_dist(sph_cart_array(B),sph_cart_array(u))

print(calc_rhomboid(2.5,4.*math.sin(math.pi*2./5.),4.))

#h_base_iso = math.sqrt(r*r - d_isoscele*d_isoscele/4.)
ext_isoscele = calc_rhomboid(h,r,d_isoscele) #d_isoscele + 2.*((d_isoscele/2.)/
 ↪h_base_iso)*h
#h_base_equ = math.sqrt(r*r - d_equidistant*d_equidistant/4.)
ext_equidistant = calc_rhomboid(h,r,d_equidistant) #d_equidistant + 2.
 ↪*((d_equidistant/2.)/h_base_equ)*h
```

```
7.0901699437494745
```

[15]:
```
from IPython.display import Markdown as md

displaytext = """# Results

For a side length of **{:.3f}cm** centimeters of the base
20-faced icosahedron a radius of **{:.3f}cm** result

### Triangles

* the isocele triangle (60 copies) has two sides of length {:.3f}cm and one side␣
 ↪of length {:.3f}cm
* the equidistant triangle (20 copies) has a side length of {:.3f}cm

### Rhomboids

Given a defined height of {:.3f}cm the Rhomboids have sidelength of
* {:.3f}cm and {:.3f}cm for the ones extending the equidistant triangles (60␣
 ↪copies)
* {:.3f}cm and {:.3f}cm for the ones extending the isoscele triangles (60 copies)

""".
 ↪format(a,r,d_isoscele,d_equidistant,d_equidistant,h,d_equidistant,ext_equidistant,d_isoscele,
md(displaytext)
```

[15]:

1 Results

For a side length of **4.000cm** centimeters of the base 20-faced icosahedron a radius of **3.804cm** result

1.0.1 Triangles

- the isocele triangle (60 copies) has two sides of length 2.079cm and one side of length 2.351cm
- the equidistant triangle (20 copies) has a side length of 2.351cm

1.0.2 Rhomboids

Given a defined height of 2.500cm the Rhomboids have sidelength of * 2.351cm and 3.976cm for the ones extending the equidistant triangles (60 copies) * 2.079cm and 3.500cm for the ones extending the isoscele triangles (60 copies)

[16]:
```
(r+h)/math.sin(alpha)
```

[16]: 6.628655560595669

[ ]:
```

```