Well Geometry

Robert Atkinson
04 Oct 2019

We explore the geometry of various labware.

Basics

```
(*FrontEndExecute[{FrontEndToken[InputNotebook[],"SelectAll"]}];
FrontEndExecute[{FrontEndToken[InputNotebook[],"SelectionOpenAllGroups"]}];*)
```

```
On[Assert]
assert[expr_] := Module[{value = Evaluate[expr]},
  If[BooleanQ[value], Assert[value, HoldForm[expr]]]
 ]
SetAttributes[assert, HoldAll]
```

```
printCell[cell_] := CellPrint[ExpressionCell[cell, "Output"]]
```

```
test[expr_] := Module[{evald},
  evald = Evaluate[expr];
  printCell[HoldForm[expr] → evald];
  evald]
SetAttributes[test, HoldAll]
test[7 !];
% + 1
```

```
7 ! → 5040
```

```
5041
```

```
complement[angle_] := π / 2 - angle
```

```
Clear[hasImaginary]
hasImaginary[expr_] := Module[{result},
  (*result = Reap[Scan[Function[ee, If[ee ≠ Conjugate[ee], Sow[True]]],{expr}, {-1, Infinity}]];*)
  result = Scan[Function[ee, If[ee ≠ Conjugate[ee], Return[True]]], {expr}, {-1, Infinity}];
  (*Length @ result[[2]] > 0 *)
  result === True]
SetAttributes[hasImaginary, HoldAll]
test @ hasImaginary[1 + 2 I];
test @ hasImaginary[30 !];
```

```
hasImaginary[1 + 2 i] → True
```

```
hasImaginary[30 !] → False
```

```
toDeg[rad_] := rad / Pi * 180
toRadian[deg_] := deg / 180 * Pi
```

Cone

Accessing

```
assumptions[cone[h_, r_]] := h >= 0 && r >= 0
assumptions[cone[h_, α_, apexangle]] := FullSimplify[h >= 0 && α > 0 && α < π / 2]
assumptions[cone[h_, β_, baseangle]] := FullSimplify[assumptions[cone[h, complement[β], apexangle]]]
```

```
test @ assumptions[cone[h, α, apexangle]];
test @ assumptions[cone[h, β, baseangle]];
```

```
assumptions[cone[h, α, apexangle]] → h ≥ 0&&α > 0&&2 α < π
```

```
assumptions[cone[h, β, baseangle]] → h ≥ 0&&2 β < π&&β > 0
```

```
radius[c : cone[h_, r_]] := r
radius[c : cone[h_, α_, apexangle]] := h Tan[α]
radius[c : cone[h_, β_, baseangle]] := h Cot[β]

height[c : cone[h_, r_]] := h
height[c : cone[h_, α_, apexangle]] := h
height[c : cone[h_, β_, baseangle]] := h
```

```
apexangle[c : cone[h_, r_]] := Assuming[assumptions[c], ArcTan[h, r]]
apexangle[c : cone[h_, α_, apexangle]] := α
apexangle[c : cone[h_, β_, baseangle]] := complement[baseangle[c]]
baseangle[c : cone[h_, r_]] := Assuming[assumptions[c], ArcTan[r, h]]
baseangle[c : cone[h_, α_, apexangle]] := complement[α]
baseangle[c : cone[h_, β_, baseangle]] := β
```

```
test @ apexangle[cone[h, r]];
test @ apexangle[cone[h, α, apexangle]];
test @ apexangle[cone[h, β, baseangle]];
test @ baseangle[cone[h, r]];
test @ baseangle[cone[h, α, apexangle]];
test @ baseangle[cone[h, β, baseangle]];
```

```
apexangle[cone[h, r]] → ArcTan[h, r]
```

```
apexangle[cone[h, α, apexangle]] → α
```

apexangle[cone[h, β, baseangle]] → $\frac{\pi}{2} - \beta$

```
baseangle[cone[h, r]] → ArcTan[r, h]
```

baseangle[cone[h, α, apexangle]] → $\frac{\pi}{2} - \alpha$

```
baseangle[cone[h, β, baseangle]] → β
```

Conversion

```
toCone[c : cone[h_, r_]] := c
toCone[c : cone[h_, α_, apexangle]] := cone[h, radius[c]]
toCone[c : cone[h_, β_, baseangle]] := cone[h, radius[c]]

toCartesian[c : cone[h_, r_]] := toCone @ c
toCartesian[c : cone[h_, α_, apexangle]] := toCone @ c
toCartesian[c : cone[h_, β_, baseangle]] := toCone @ c

toApexAngled[c : cone[h_, r_]] := cone[h, apexangle[c], apexangle]
toApexAngled[c : cone[h_, α_, apexangle]] := c
toApexAngled[c : cone[h_, β_, baseangle]] := cone[h, apexangle[c], apexangle]

toBaseAngled[c : cone[h_, r_]] := cone[h, baseangle[c], baseangle]
toBaseAngled[c : cone[h_, α_, apexangle]] := cone[h, baseangle[c], baseangle]
toBaseAngled[c : cone[h_, β_, baseangle]] := c

scaled[c : cone[h_, r_], factor_] := cone[h * factor, r * factor]
scaled[c : cone[h_, α_, apexangle], factor_] := toApexAngled @ scaled[toCartesian @ c, factor]
scaled[c : cone[h_, β_, baseangle], factor_] := toBaseAngled @ scaled[toCartesian @ c, factor]
```

```
test @ toCone[cone[h, r]];
test @ toCone[cone[h, α, apexangle]] ;
test @ toCone[cone[h, β, baseangle]];
test @ toApexAngled[cone[h, r]];
test @ toApexAngled[cone[h, α, apexangle]];
test @ toApexAngled[cone[h, β, baseangle]];
test @ toBaseAngled[cone[h, r]];
test @ toBaseAngled[cone[h, α, apexangle]];
test @ toBaseAngled[cone[h, β, baseangle]];
test @ scaled[cone[h, r], 2];
test @ scaled[cone[h, α, apexangle], 2] ;
test @ scaled[cone[h, β, baseangle], 2];
```

toCone[cone[h, r]] → cone[h, r]

toCone[cone[h, α, apexangle]] → cone[h, h Tan[α]]

toCone[cone[h, β, baseangle]] → cone[h, h Cot[β]]

toApexAngled[cone[h, r]] → cone[h, ArcTan[h, r], apexangle]

toApexAngled[cone[h, α, apexangle]] → cone[h, α, apexangle]

toApexAngled[cone[h, β, baseangle]] → cone[h, $\frac{\pi}{2} - \beta$, apexangle]

toBaseAngled[cone[h, r]] → cone[h, ArcTan[r, h], baseangle]

toBaseAngled[cone[h, α, apexangle]] → cone[h, $\frac{\pi}{2} - \alpha$, baseangle]

toBaseAngled[cone[h, β, baseangle]] → cone[h, β, baseangle]

scaled[cone[h, r], 2] → cone[2 h, 2 r]

scaled[cone[h, α, apexangle], 2] → cone[2 h, ArcTan[2 h, 2 h Tan[α]], apexangle]

scaled[cone[h, β, baseangle], 2] → cone[2 h, ArcTan[2 h Cot[β], 2 h], baseangle]

Volume

```
volume[c : cone[h_, r_]] := Pi r r h / 3
volume[c : cone[h_, α_, apexangle]] := volume @ toCartesian @ c
volume[c : cone[h_, β_, baseangle]] := volume @ toCartesian @ c
test @ volume[cone[h, r]];
test @ volume[cone[h, α, apexangle]];
test @ volume[cone[h, β, baseangle]];
```

volume[cone[h, r]] → $\frac{1}{3} h \pi r^2$

volume[cone[h, α, apexangle]] → $\frac{1}{3} h^3 \pi \, \mathrm{Tan}[\alpha]^2$

volume[cone[h, β, baseangle]] → $\frac{1}{3} h^3 \pi \, \mathrm{Cot}[\beta]^2$

Height and Depth

Final

```
genericConeDepthFromVolume[] := Module[{c, cc, h, r, hh, vol, a, eqn, solns, soln},
  (* conjures up a soln with varaibles known to be free *)
  c = cone[h, r];
  cc = scaled[c, hh / h];
  a = assumptions[c] && assumptions[cc] && vol ≥ 0;
  eqn = FullSimplify[vol == volume[c] - volume[cc], a];
  solns = Assuming[a, Solve[eqn, hh]];
  soln = FullSimplify[h - (hh /. First @ solns), a];
  genericConeDepthFromVolume[] = {h, r, vol, soln}
 ]
test @ genericConeDepthFromVolume[];
```

genericConeDepthFromVolume[] → $\left\{\text{h\$2812, r\$2812, vol\$2812, h\$2812} - \left(\frac{\text{h\$2812}}{\text{r\$2812}}\right)^{2/3} \left(\text{h\$2812 r\$2812}^2 - \frac{3\,\text{vol\$2812}}{\pi}\right)^{1/3}\right\}$

```
depthFromVolume[c : cone[h_, r_], v_] := Module[{hh, rr, vol, soln},
  {hh, rr, vol, soln} = genericConeDepthFromVolume[];
  (soln /. {hh → h, rr → r, vol → v}) // FullSimplify
 ]
depthFromVolume[c : cone[h_, α_, apexangle], v_] := depthFromVolume[toCartesian @ c, v]
depthFromVolume[c : cone[h_, β_, baseangle], v_] := depthFromVolume[toCartesian @ c, v]

test @ depthFromVolume[cone[h, r], volume];
```

depthFromVolume[cone[h, r], volume] → $h - \left(\frac{h}{r}\right)^{2/3} \left(h\,r^2 - \frac{3\,\text{volume}}{\pi}\right)^{1/3}$

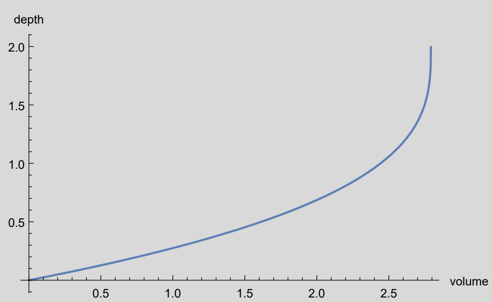
Testing

```
example = cone[2, π / 6, apexangle]
{ volume[example], volume[example] // N }
expr = test @ depthFromVolume[example, v];
Plot[expr, {v, 0, volume[example]}, AxesLabel → {"volume", "depth"}]
```

cone$\left[2, \frac{\pi}{6}, \text{apexangle}\right]$

$\left\{\frac{8\,\pi}{9}, 2.79253\right\}$

depthFromVolume[example, v] → $2 - \left(8 - \frac{9\,v}{\pi}\right)^{1/3}$



Inverted Cone

Construction & Conversion

```
toCone[c : invertedCone[h_, r_]] := invert @ c
toCone[c : invertedCone[h_, α_, apexangle]] := invert @ c
toCone[c : invertedCone[h_, β_, baseangle]] := invert @ c

toCartesian[c : invertedCone[h_, r_]] := invert @ toCartesian @ invert @ c
toCartesian[c : invertedCone[h_, α_, apexangle]] := invert @ toCartesian @ invert @ c
toCartesian[c : invertedCone[h_, β_, baseangle]] := invert @ toCartesian @ invert @ c

invert[c : invertedCone[h_, r_]] := cone[h, r]
invert[c : invertedCone[h_, α_, apexangle]] := cone[h, α, apexangle]
invert[c : invertedCone[h_, β_, baseangle]] := cone[h, β, baseangle]

invert[c : cone[h_, r_]] := invertedCone[h, r]
invert[c : cone[h_, α_, apexangle]] := invertedCone[h, α, apexangle]
invert[c : cone[h_, β_, baseangle]] := invertedCone[h, β, baseangle]

scaled[c:invertedCone[h_, r_], factor_] := invertedCone[h * factor, r * factor]
scaled[c:invertedCone[h_, α_, apexangle], factor_] := toApexAngled @ scaled[toCartesian @ c, factor]
scaled[c:invertedCone[h_, β_, baseangle], factor_] := toBaseAngled @ scaled[toCartesian @ c, factor]
```

```
test @ scaled[invertedCone[h, r], 2];
test @ scaled[invertedCone[h, α, apexangle], 2] ;
test @ scaled[invertedCone[h, β, baseangle], 2];
```

```
scaled[invertedCone[h, r], 2] → invertedCone[2 h, 2 r]
```

```
scaled[invertedCone[h, α, apexangle], 2] → toApexAngled[invertedCone[2 h, 2 h Tan[α]]]
```

```
scaled[invertedCone[h, β, baseangle], 2] → toBaseAngled[invertedCone[2 h, 2 h Cot[β]]]
```

Accessing

```
assumptions[c : invertedCone[h_, r_]] := assumptions[toCone @ c]
assumptions[c : invertedCone[h_, α_, apexangle]] := assumptions[toCone @ c]
assumptions[c : invertedCone[h_, β_, baseangle]] := assumptions[toCone @ c]
test @ assumptions[invertedCone[h, α, apexangle]];
test @ assumptions[invertedCone[h, β, baseangle]];
```

```
assumptions[invertedCone[h, α, apexangle]] → h ≥ 0&&α > 0&&2 α < π
```

```
assumptions[invertedCone[h, β, baseangle]] → h ≥ 0&&2 β < π&&β > 0
```

```
radius[c : invertedCone[h_, r_]] := r
radius[c : invertedCone[h_, α_, apexangle]] := radius @ invert @ c
radius[c : invertedCone[h_, β_, baseangle]] := radius @ invert @ c

height[c:invertedCone[h_, r_]] := h
height[c:invertedCone[h_, α_, apexangle]] := h
height[c:invertedCone[h_, β_, baseangle]] := h
```

```
apexangle[c : invertedCone[h_, r_]] := Assuming[assumptions[c], ArcTan[h, r]]
apexangle[c : invertedCone[h_, α_, apexangle]] := α
apexangle[c : invertedCone[h_, β_, baseangle]] := complement[baseangle[c]]
baseangle[c : invertedCone[h_, r_]] := Assuming[assumptions[c], ArcTan[r, h]]
baseangle[c : invertedCone[h_, α_, apexangle]] := complement[α]
baseangle[c : invertedCone[h_, β_, baseangle]] := β
```

```
test @ apexangle[invertedCone[h, r]];
test @ apexangle[invertedCone[h, α, apexangle]];
test @ apexangle[invertedCone[h, β, baseangle]];
test @ baseangle[invertedCone[h, r]];
test @ baseangle[invertedCone[h, α, apexangle]];
test @ baseangle[invertedCone[h, β, baseangle]];
```

apexangle[invertedCone[h, r]] → ArcTan[h, r]

apexangle[invertedCone[h, α, apexangle]] → α

apexangle[invertedCone[h, β, baseangle]] → $\frac{\pi}{2} - \beta$

baseangle[invertedCone[h, r]] → ArcTan[r, h]

baseangle[invertedCone[h, α, apexangle]] → $\frac{\pi}{2} - \alpha$

baseangle[invertedCone[h, β, baseangle]] → β

Conversion Redux

```
toInvertedCone[c : invertedCone[h_, r_]] := c
toInvertedCone[c : invertedCone[h_, α_, apexangle]] := invertedCone[h, h Tan[α]]
toInvertedCone[c : invertedCone[h_, β_, baseangle]] := toInvertedCone[toApexAngled[c]]

toCartesian[c : invertedCone[h_, r_]] := toInvertedCone @ c
toCartesian[c : invertedCone[h_, α_, apexangle]] := toInvertedCone @ c
toCartesian[c : invertedCone[h_, β_, baseangle]] := toInvertedCone @ c

toApexAngled[c : invertedCone[h_, r_]] := invertedCone[h, apexangle[c], apexangle]
toApexAngled[c : invertedCone[h_, α_, apexangle]] := c
toApexAngled[c : invertedCone[h_, β_, baseangle]] := invertedCone[h, apexangle[c], apexangle]

toBaseAngled[c : invertedCone[h_, r_]] := invertedCone[h, baseangle[c], baseangle]
toBaseAngled[c : invertedCone[h_, α_, apexangle]] := invertedCone[h, baseangle[c], baseangle]
toBaseAngled[c : invertedCone[h_, β_, baseangle]] := c
```

```
test @ toInvertedCone[invertedCone[h, r]];
test @ toInvertedCone[invertedCone[h, α, apexangle]];
test @ toInvertedCone[invertedCone[h, β, baseangle]];
test @ toApexAngled[invertedCone[h, r]];
test @ toApexAngled[invertedCone[h, α, apexangle]];
test @ toApexAngled[invertedCone[h, β, baseangle]];
test @ toBaseAngled[invertedCone[h, r]];
test @ toBaseAngled[invertedCone[h, α, apexangle]];
test @ toBaseAngled[invertedCone[h, β, baseangle]];
```

toInvertedCone[invertedCone[h, r]] → invertedCone[h, r]

toInvertedCone[invertedCone[h, α, apexangle]] → invertedCone[h, h Tan[α]]

toInvertedCone[invertedCone[h, β, baseangle]] → invertedCone[h, h Cot[β]]

toApexAngled[invertedCone[h, r]] → invertedCone[h, ArcTan[h, r], apexangle]

toApexAngled[invertedCone[h, α, apexangle]] → invertedCone[h, α, apexangle]

toApexAngled[invertedCone[h, β, baseangle]] → invertedCone$\left[h, \frac{\pi}{2} - \beta, \text{apexangle}\right]$

toBaseAngled[invertedCone[h, r]] → invertedCone[h, ArcTan[r, h], baseangle]

toBaseAngled[invertedCone[h, α, apexangle]] → invertedCone$\left[h, \frac{\pi}{2} - \alpha, \text{baseangle}\right]$

toBaseAngled[invertedCone[h, β, baseangle]] → invertedCone[h, β, baseangle]

Volume

```
volume[c : invertedCone[h_, r_]] := volume @ toCone @ c
volume[c : invertedCone[h_, α_, apexangle]] := volume @ toCone @ c
volume[c : invertedCone[h_, β_, baseangle]] := volume @ toCone @ c
test @ volume[invertedCone[h, r]];
test @ volume[invertedCone[h, α, apexangle]];
test @ volume[invertedCone[h, β, baseangle]];
```

volume[invertedCone[h, r]] → $\frac{1}{3} h \pi r^2$

volume[invertedCone[h, α, apexangle]] → $\frac{1}{3} h^3 \pi \operatorname{Tan}[\alpha]^2$

volume[invertedCone[h, β, baseangle]] → $\frac{1}{3} h^3 \pi \operatorname{Cot}[\beta]^2$

Height and Depth

Final

```
genericInvertedConeDepthFromVolume[] := Module[{c, h, α, hh, vol, a, eqn, solns, soln},
  c = invertedCone[h, α, apexangle];
  a = assumptions[c] && vol ≥ 0 ;
  eqn = FullSimplify[vol == volume[c], a];
  solns = Assuming[a, Solve[eqn, h]];
  soln = FullSimplify[h /. solns[[2]], a];
  genericInvertedConeDepthFromVolume[] = {α, vol, soln}
 ]
test @ genericInvertedConeDepthFromVolume[];
```

genericInvertedConeDepthFromVolume[] → $\left\{\alpha\$3751,\ vol\$3751,\ \left(\frac{3}{\pi}\right)^{1/3}\left(vol\$3751\ \mathrm{Cot}[\alpha\$3751]^2\right)^{1/3}\right\}$

```
depthFromVolume[c : invertedCone[ignored_, α_, apexangle], v_] := Module[{αα, vol, soln},
  {αα, vol, soln} = genericInvertedConeDepthFromVolume[];
  (soln /. {αα → α, vol → v}) // FullSimplify
 ]
depthFromVolume[c : invertedCone[h_, r_], v_] := depthFromVolume[toApexAngled @ c, v]
depthFromVolume[c : invertedCone[h_, β_, baseangle], v_] := depthFromVolume[toApexAngled @ c, v]

test @ depthFromVolume[invertedCone[ignored, α, apexangle], volume];
test @ depthFromVolume[invertedCone[h, r], volume];
test @ depthFromVolume[invertedCone[h, β, baseangle], volume];
```

depthFromVolume[invertedCone[ignored, α, apexangle], volume] → $\left(\frac{3}{\pi}\right)^{1/3}\left(\mathrm{volume}\ \mathrm{Cot}[\alpha]^2\right)^{1/3}$

depthFromVolume[invertedCone[h, r], volume] → $\left(\frac{3}{\pi}\right)^{1/3}\left(\frac{h^2\ \mathrm{volume}}{r^2}\right)^{1/3}$

depthFromVolume[invertedCone[h, β, baseangle], volume] → $\left(\frac{3}{\pi}\right)^{1/3}\left(\mathrm{volume}\ \mathrm{Tan}[\beta]^2\right)^{1/3}$

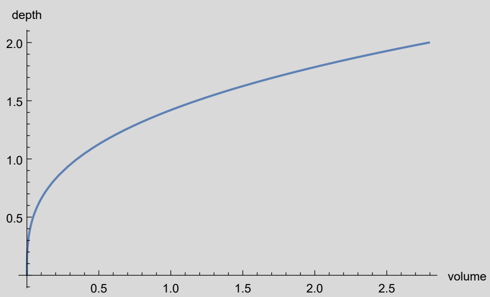
Testing

```
example = invertedCone[2, π / 6, apexangle]
{ volume[example], volume[example] // N }
expr = test @ depthFromVolume[example, v];
Plot[expr, {v, 0, volume[example]}, AxesLabel → {"volume", "depth"}]
```

invertedCone$\left[2, \frac{\pi}{6}, \text{apexangle}\right]$

$\left\{\frac{8\pi}{9}, 2.79253\right\}$

depthFromVolume[example, v] → $\frac{3^{2/3}\, v^{1/3}}{\pi^{1/3}}$



Cylinder

Accessing

```
assumptions[cylinder[h_, r_]] := h >= 0 && r >= 0
```

```
test @ assumptions[cylinder[h, r]];
```

assumptions[cylinder[h, r]] → h ≥ 0 && r ≥ 0

```
emptyCylinder[] := cylinder[0, 0]
height[c : cylinder[h_, r_]] := h
radius[c : cylinder[h_, r_]] := r
```

```
toCartesian[c : cylinder[h_, r_]] := c
toApexAngled[c : cylinder[h_, r_]] := c
toBaseAngled[c : cylinder[h_, r_]] := c
```

Volume

```
volume[cylinder[h_, r_]] := Pi r r h
test @ volume[cylinder[h, r]];
test @ volume @ emptyCylinder[];
```

volume[cylinder[h, r]] → $h\,\pi\,r^2$

volume[emptyCylinder[]] → 0

Height and Depth

Final

```
depthFromVolume[c : cylinder[_, 0], v_] := 0
depthFromVolume[c : cylinder[0, _], v_] := 0
depthFromVolume[c : cylinder[_, r_], v_] := Module[{hh}, hh /. First @ Solve[v == volume[cylinder[hh, r]], hh]]
test @ depthFromVolume[cylinder[ignored, r], volume];
test @ depthFromVolume[cylinder[1, 2], volume];
test @ depthFromVolume[emptyCylinder[], volume];
```

depthFromVolume[cylinder[ignored, r], volume] → $\frac{\text{volume}}{\pi r^2}$

depthFromVolume[cylinder[1, 2], volume] → $\frac{\text{volume}}{4\pi}$

depthFromVolume[emptyCylinder[], volume] → 0

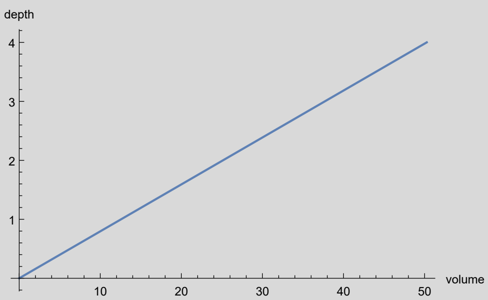
Testing

```
example = cylinder[4, 2]
{ volume[example], volume[example] // N }
expr = test @ depthFromVolume[example, v];
Plot[expr, {v, 0, volume[example]}, AxesLabel → {"volume", "depth"}]
```

cylinder[4, 2]

{16 π, 50.2655}

depthFromVolume[example, v] → $\frac{v}{4\pi}$



Right Conical Frustum

Accessing

```
assumptions[frustum[h_, rbig_, rsmall_]] := h ≥ 0 && rbig ≥ 0 && rsmall ≥ 0 && rbig > rsmall
assumptions[frustum[h_, rbig_, α_, apexangle]] := FullSimplify @ assumptions[frustum[h, rbig, complement[α], baseangle]]
assumptions[frustum[h_, rbig_, β_, baseangle]] := FullSimplify[h ≥ 0 && rbig ≥ 0 && β > 0 && β < π / 2]
```

```
test @ assumptions[frustum[h, rbig, α, apexangle]];
test @ assumptions[frustum[h, rbig, β, baseangle]];
```

```
assumptions[frustum[h, rbig, α, apexangle]] → h ≥ 0 && rbig ≥ 0 && 2 α < π && α > 0
```

```
assumptions[frustum[h, rbig, β, baseangle]] → h ≥ 0 && rbig ≥ 0 && β > 0 && 2 β < π
```

```
apexangle[f : frustum[h_, rbig_, α_, apexangle]] := α
apexangle[f : frustum[h_, rbig_, β_, baseangle]] := complement[baseangle[f]]
apexangle[f : frustum[h_, rbig_, rsmall_]] := Assuming[assumptions[f], ArcTan[h, rbig - rsmall]]

baseangle[f : frustum[h_, rbig_, α_, apexangle]] := complement[ apexangle[f]]
baseangle[f : frustum[h_, rbig_, β_, baseangle]] := β
baseangle[f : frustum[h_, rbig_, rsmall_]] := Assuming[assumptions[f], ArcTan[rbig - rsmall, h]]

baseangle[f : frustum[h_, rbig_, rbig_ - h_ Cot[β_]]] := β

test @ apexangle[frustum[h, rbig, rsmall]];
test @ baseangle[frustum[h, rbig, rsmall]];
test @ { baseangle[frustum[1, 3, 2]], baseangle[frustum[Sqrt[3], 2, 1]]};
```

```
apexangle[frustum[h, rbig, rsmall]] → ArcTan[h, rbig - rsmall]
```

```
baseangle[frustum[h, rbig, rsmall]] → ArcTan[rbig - rsmall, h]
```

$\{\text{baseangle}[\text{frustum}[1, 3, 2]], \text{baseangle}[\text{frustum}[\sqrt{3}, 2, 1]]\} \to \{\frac{\pi}{4}, \frac{\pi}{3}\}$

```
Solve[(rbig - rsmall) / h == Tan[α], rsmall]
Solve[(rbig - rsmall) / h == Tan[α], rbig]
```

```
{{rsmall → rbig - h Tan[α]}}
```

```
{{rbig → rsmall + h Tan[α]}}
```

```
rbig[h_, rsmall_, α_, apexangle] := rsmall + h Tan[α]
rsmall[h_, rbig_, α_, apexangle] := rbig - h Tan[α]
rbig[h_, rsmall_, β_, baseangle] := rbig[h, rsmall, complement[β], apexangle]
rsmall[h_, rbig_, β_, baseangle] := rsmall[h, rsmall, complement[β], apexangle]
```

```
height[f : frustum[h_, rbig_, α_, apexangle]] := h
height[f : frustum[h_, rbig_, β_, baseangle]] := h
height[f : frustum[h_, rbig_, rsmall_]] := h
```

```
rbig[f : frustum[h_, rbig_, α_, apexangle]] := rbig
rbig[f : frustum[h_, rbig_, β_, baseangle]] := rbig
rbig[f : frustum[h_, rbig_, rsmall_]] := rbig
```

```
Tan[α] / Cot[complement[α]] == 1
rsmall[f:frustum[h_, rbig_, α_, apexangle]] := Assuming[assumptions[f], rsmall[h, rbig, α, apexangle]]
rsmall[f:frustum[h_, rbig_, β_, baseangle]] := Assuming[assumptions[f], rsmall[h, rbig, β, baseangle]]
rsmall[f:frustum[h_, rbig_, rsmall_]] := rsmall
rsmall[f:frustum[h_, rbig_, ArcTan[rbig_ - rsmall_, h_], baseangle]] := rsmall
test @ rsmall[frustum[h, rbig, α, apexangle]];
test @ rsmall[frustum[h, rbig, β, baseangle]];
test @ rsmall[frustum[h, rbig, rsmall]];
```

```
True
```

```
rsmall[frustum[h, rbig, α, apexangle]] → rbig - h Tan[α]
```

```
rsmall[frustum[h, rbig, β, baseangle]] → rsmall - h Cot[β]
```

```
rsmall[frustum[h, rbig, rsmall]] → rsmall
```

Construction & Conversion

```
toFrustum[f: frustum[h_, rbig_, α_, apexangle]] := frustum[h, rbig, rsmall[f]]
toFrustum[f: frustum[h_, rbig_, β_, baseangle]] := frustum[h, rbig, rsmall[f]]
toFrustum[f: frustum[h_, rbig_, rsmall_]] := f

toCartesian[f: frustum[h_, rbig_, α_, apexangle]] := toFrustum @ f
toCartesian[f: frustum[h_, rbig_, β_, baseangle]] := toFrustum @ f
toCartesian[f: frustum[h_, rbig_, rsmall_]] := toFrustum @ f

toApexAngled[f:frustum[h_, rbig_, α_, apexangle]] := f
toApexAngled[f:frustum[h_, rbig_, β_, baseangle]] := frustum[h, rbig, complement[β], apexangle]
toApexAngled[f:frustum[h_, rbig_, rsmall_]] := frustum[h, rbig, apexangle[f], apexangle]

toBaseAngled[f:frustum[h_, rbig_, α_, apexangle]] := frustum[h, rbig, complement[α], baseangle]
toBaseAngled[f:frustum[h_, rbig_, β_, baseangle]] := f
toBaseAngled[f:frustum[h_, rbig_, rsmall_]] := frustum[h, rbig, baseangle[f], baseangle]
```

```
test @ toCartesian @ frustum[h, rbig, β, baseangle];
test @ toBaseAngled @ %;
test @ toApexAngled @ %%;
test @ toFrustum @ %;
test @ toBaseAngled @ %%;
```

```
toCartesian[frustum[h, rbig, β, baseangle]] → frustum[h, rbig, rsmall - h Cot[β]]
```

```
toBaseAngled[%] → frustum[h, rbig, ArcTan[rbig - rsmall + h Cot[β], h], baseangle]
```

```
toApexAngled[%%] → frustum[h, rbig, ArcTan[h, rbig - rsmall + h Cot[β]], apexangle]
```

```
toFrustum[%] → frustum[h, rbig, rsmall - h Cot[β]]
```

toBaseAngled[%%] → frustum$\left[\text{h, rbig, } \frac{\pi}{2} - \text{ArcTan[h, rbig} - \text{rsmall} + \text{h Cot}[\beta]], \text{ baseangle}\right]$

```
test @ toBaseAngled @ frustum[h, rbig, rsmall];
test @ toCartesian @ %;
```

```
toBaseAngled[frustum[h, rbig, rsmall]] → frustum[h, rbig, ArcTan[rbig - rsmall, h], baseangle]
```

```
toCartesian[%] → frustum[h, rbig, rsmall]
```

Volume

```
genericConeHeightCartesianFrustum[] := Module[{f, h, rbig, rsmall, eqn, ch},
  f = frustum[h, rbig, rsmall];
  eqn = ch / rbig == h / (rbig - rsmall);
  genericConeHeightCartesianFrustum[] = {h, rbig, rsmall, ch /. First @ Solve[eqn, ch]}
 ]

coneHeight[f:frustum[h_, rbig_, α_, apexangle]] := rbig / Tan[α]
coneHeight[f:frustum[h_, rbig_, β_, baseangle]] := rbig / Cot[β]
coneHeight[f:frustum[h_, rbig_, rsmall_]] := Module[{hh, rrbig, rrsmall, ch},
  {hh, rrbig, rrsmall, ch} = genericConeHeightCartesianFrustum[];
  ch /. {hh → h, rrbig → rbig, rrsmall → rsmall}
 ]
test @ coneHeight[frustum[h, rbig, α, apexangle]];
test @ coneHeight[frustum[h, rbig, β, baseangle]];
test @ toApexAngled @ frustum[h, rbig, β, baseangle];
test @ coneHeight @ %;
test @ coneHeight[frustum[h, rbig, rsmall]];
test @ coneHeight[frustum[1, 3, 2]];
```

coneHeight[frustum[h, rbig, α, apexangle]] → rbig Cot[α]

coneHeight[frustum[h, rbig, β, baseangle]] → rbig Tan[β]

toApexAngled[frustum[h, rbig, β, baseangle]] → frustum$\left[h, rbig, \frac{\pi}{2} - \beta, apexangle\right]$

coneHeight[%] → rbig Tan[β]

coneHeight[frustum[h, rbig, rsmall]] → $\frac{h\ rbig}{rbig - rsmall}$

coneHeight[frustum[1, 3, 2]] → 3

```
fullCone[f: frustum[h_, rbig_, α_, apexangle]] := cone[coneHeight[f], α, apexangle]
fullCone[f: frustum[h_, rbig_, β_, baseangle]] := fullCone @ toApexAngled @ f
fullCone[f: frustum[h_, rbig_, rsmall_]] := cone[coneHeight[f], rbig]
```

```
topCone[f: frustum[h_, rbig_, α_, apexangle]] := cone[coneHeight[f] - h, α, apexangle]
topCone[f: frustum[h_, rbig_, β_, baseangle]] := topCone @ toApexAngled @ f
topCone[f: frustum[h_, rbig_, rsmall_]] := Module[{full, eqn, scale, result},
  full = fullCone[f];
  result = scaled[full, scale];
  eqn = radius[result] == rsmall;
  result /. First @ Solve[eqn, scale]
 ]
test @ topCone[frustum[h, rbig, rsmall]];
```

topCone[frustum[h, rbig, rsmall]] → cone$\left[\frac{h\ rsmall}{rbig - rsmall}, rsmall\right]$

```
volume[f: frustum[h_, rbig_, rsmall_]] := volume[fullCone[f]] - volume[topCone[f]] // FullSimplify
volume[f: frustum[h_, rbig_, α_, apexangle]] := volume[fullCone[f]] - volume[topCone[f]] // FullSimplify
volume[f: frustum[h_, rbig_, β_, baseangle]] := volume @ toApexAngled[f]
```

```
(* compare to textbook answer 1/3 h π (r1^2+r1 r2+r2^2) *)
test @ volume[frustum[h, r1, r2]];
test @ volume[frustum[h, r, α, apexangle]];
test @ volume[toFrustum @ frustum[h, r, α, apexangle]];
% / %% // FullSimplify
test @ volume[frustum[h, r, β, baseangle]];
```

volume[frustum[h, r1, r2]] → $\frac{1}{3} h \pi \left(r1^2 + r1\, r2 + r2^2\right)$

volume[frustum[h, r, α, apexangle]] → $\frac{1}{3} h \pi \left(3 r^2 + h \operatorname{Tan}[\alpha] \left(-3 r + h \operatorname{Tan}[\alpha]\right)\right)$

volume[toFrustum[frustum[h, r, α, apexangle]]] → $\frac{1}{3} \pi \operatorname{Cot}[\alpha] \left(r^3 - \left(r - h \operatorname{Tan}[\alpha]\right)^3\right)$

1

volume[frustum[h, r, β, baseangle]] → $\frac{1}{3} h \pi \left(3 r^2 + h \operatorname{Cot}[\beta] \left(-3 r + h \operatorname{Cot}[\beta]\right)\right)$

Height and Depth

Experimenting

In the below, the ‘Solve’ calls generate three solutions each. Which index to choose is unfortunately data-dependent.

```
depthFromVolumeExperiment[f : frustum[h_, rbig_, rsmall_], vol_, index_] := Module[{hh, ff, eqn, solns},
  (* we're looking for a frustum with same base angle and bottom radius, but different height *)
  ff = frustum[hh, rbig, baseangle[f], baseangle];
  eqn = FullSimplify[vol == volume[ff], assumptions[f] && vol ≥ 0];
  solns = Solve[eqn, hh];
  FullSimplify[hh /. solns[[index]], assumptions[f] && vol ≥ 0]
 ]
depthFromVolumeExperiment[f : frustum[h_, rbig_, rsmall_], vol_] := depthFromVolumeExperiment[f, vol, 1]
test @ depthFromVolumeExperiment[frustum[h, r1, r2], vol];
```

depthFromVolumeExperiment[frustum[h, r1, r2], vol] → $\dfrac{h\, r1 + \dfrac{\left(-h^2 \left(h \pi\, r1^3 + 3 \left(-r1 + r2\right) vol\right)\right)^{1/3}}{\pi^{1/3}}}{r1 - r2}$

```
depthFromVolumeExperiment[f : frustum[ignored_, rbig_, α_, apexangle], vol_, index_] := Module[{hh, ff, eqn, solns},
  (* we're looking for a frustum with same base angle and bottom radius, but different height *)
  ff = frustum[hh, rbig, baseangle[f], baseangle];
  eqn = FullSimplify[vol == volume[ff], assumptions[f] && vol ≥ 0];
  solns = Solve[eqn, hh];
  FullSimplify[hh /. solns[[index]], assumptions[f] && vol ≥ 0]
 ]
depthFromVolumeExperiment[f : frustum[ignored_, rbig_, α_, apexangle], vol_] := depthFromVolumeExperiment[f, vol, 1]
test @ depthFromVolumeExperiment[frustum[h, r, α, apexangle], vol];
```

depthFromVolumeExperiment[frustum[h, r, α, apexangle], vol] → $\operatorname{Cot}[\alpha] \left(r - \left(r^3 - \frac{3\, vol \operatorname{Tan}[\alpha]}{\pi}\right)^{1/3}\right)$

```
depthFromVolumeExperiment[f : frustum[ignored_, rbig_, β_, baseangle], vol_, index_] := Module[{hh, ff, eqn, solns},
  (* we're looking for a frustum with same base angle and bottom radius, but different height *)
  ff = frustum[hh, rbig, baseangle[f], baseangle];
  eqn = FullSimplify[vol == volume[ff], assumptions[f] && vol ≥ 0];
  solns = Solve[eqn, hh];
  FullSimplify[hh /. solns[[index]], assumptions[f] && vol ≥ 0]
 ]
depthFromVolumeExperiment[f : frustum[ignored_, rbig_, β_, baseangle], vol_] := depthFromVolumeExperiment[f, vol, 1]
test @ depthFromVolumeExperiment[frustum[h, r, β, baseangle], vol];
```

depthFromVolumeExperiment[frustum[h, r, β, baseangle], vol] → $\left(r - \left(r^3 - \frac{3\,\text{vol}\,\text{Cot}[\beta]}{\pi}\right)^{1/3}\right)\text{Tan}[\beta]$

Final Angled

```
genericFrustumDepthFromVolumeApex[] := Module[{f, h, rbig, α, vol, a, eqn, solns, depth},
  (* conjures up a soln with varaibles known to be free *)
  f = frustum[h, rbig, α, apexangle];
  a = assumptions[f] && vol ≥ 0;
  eqn = FullSimplify[vol == volume[f], a];
  solns = Assuming[a, Solve[eqn, h]];
  depth = FullSimplify[h /. First @ solns, a];
  genericFrustumDepthFromVolume1[] = {h, rbig, α, vol, depth}
 ]
test @ genericFrustumDepthFromVolumeApex[];
```

genericFrustumDepthFromVolumeApex[] →

$\left\{\text{h\$8602}, \text{rbig\$8602}, \alpha\text{\$8602}, \text{vol\$8602}, \text{Cot}[\alpha\text{\$8602}]\left(\text{rbig\$8602} - \left(\text{rbig\$8602}^3 - \frac{3\,\text{vol\$8602}\,\text{Tan}[\alpha\text{\$8602}]}{\pi}\right)^{1/3}\right)\right\}$

```
depthFromVolume[f : frustum[ignored_, rbig_, α_, apexangle], vol_] := Module[{hh, rr, αα, vv, eqn, depth},
  {hh, rr, αα, vv, depth} = genericFrustumDepthFromVolumeApex[];
  depth /. {rr → rbig, αα → α, vv → vol}
 ]
generalApexFrustum = frustum[h, rbig, α, apexangle]
test @ depthFromVolume[generalApexFrustum, vol];
```

frustum[h, rbig, α, apexangle]

depthFromVolume[generalApexFrustum, vol] → $\text{Cot}[\alpha]\left(\text{rbig} - \left(\text{rbig}^3 - \frac{3\,\text{vol}\,\text{Tan}[\alpha]}{\pi}\right)^{1/3}\right)$

```
depthFromVolume[f : frustum[ignored_, rbig_, β_, baseangle], vol_] := Module[{hh, rr, αα, vv, eqn, soln},
  {hh, rr, αα, vv, soln} = genericFrustumDepthFromVolumeApex[];
  soln /. {rr → rbig, αα → apexangle[f], vv → vol}
 ]
generalBaseFrustum = frustum[h, rbig, β, baseangle]
test @ depthFromVolume[generalBaseFrustum, vol];
```

frustum[h, rbig, β, baseangle]

depthFromVolume[generalBaseFrustum, vol] → $\left(\text{rbig} - \left(\text{rbig}^3 - \frac{3\,\text{vol}\,\text{Cot}[\beta]}{\pi}\right)^{1/3}\right)\text{Tan}[\beta]$

Final Cartesian

```
genericFrustumDepthFromVolumeCartesian[] :=
 Module[{f, ch, fullf, topf, scaledTop, scale, h, rbig, rsmall, vol, a, eqn, solns, soln, depth},
  f = frustum[h, rbig, rsmall];
  fullf = fullCone[f];
  topf = topCone[f];
  scaledTop = scaled[topf, scale];
  a = assumptions[fullf] && assumptions[scaledTop] && vol ≥ 0;
  eqn = (volume[fullf] - volume[scaledTop]) == vol;
  solns = Assuming[a, Solve[eqn, scale]];
  soln = solns[[2]];
  depth = FullSimplify[(height[fullf] - height[scaledTop]) /. soln , a];
  genericFrustumDepthFromVolumeCartesian[] = { h, rbig, rsmall, vol, depth }
 ]
test @ genericFrustumDepthFromVolumeCartesian[];
```

genericFrustumDepthFromVolumeCartesian[] →

$$\left\{h\$11876,\ rbig\$11876,\ rsmall\$11876,\ vol\$11876,\ \frac{h\$11876\ rbig\$11876 - h\$11876^{2/3}\left(h\$11876\ rbig\$11876^3 + \frac{3\,(-rbig\$11876+rsmall\$11876)\ vol\$11876}{\pi}\right)^{1/3}}{rbig\$11876 - rsmall\$11876}\right\}$$

We compute depth from volume two different ways, then show they’re the same. We then choose for use the version that avoids trigonometry (in the apex-angled conversion).

```
depthFromVolume1[f : frustum[ignored_, rbig_, rsmall_], vol_] := Module[{hh, rr, αα, vv, eqn, depth},
  {hh, rr, αα, vv, depth} = genericFrustumDepthFromVolumeApex[];
  depth /. {rr → rbig, αα → apexangle[f], vv → vol}
 ]
depthFromVolume2[f : frustum[h_, rbig_, rsmall_], vol_] := Module[{hh, rrbig, rrsmall, vv, eqn, depth},
  { hh, rrbig, rrsmall, vv, depth } = genericFrustumDepthFromVolumeCartesian[];
  depth /. {hh → h, rrbig → rbig, rrsmall → rsmall, vv → vol }
 ]
generalFrustum = frustum[h, rbig, rsmall]
test @ depthFromVolume1[generalFrustum, vol];
test @ depthFromVolume2[generalFrustum, vol];
Module[{d = (rbig - rsmall), r1 = %%, r2 = %, fn, rules},
 rules = {rbig^3 → t1, (rbig - rsmall) → t2, (-rbig + rsmall) → -t2, -3 t2 vol / Pi → t3};
 fn = Function[r, (((Expand[-r * d] + h rbig) //. rules) ) ^3];
 fn[r1] / fn[r2] // FullSimplify
]
depthFromVolume[f : frustum[h_, rbig_, rsmall_], vol_] := depthFromVolume2[f, vol]
```

frustum[h, rbig, rsmall]

depthFromVolume1[generalFrustum, vol] → $\dfrac{h\left(rbig - \left(rbig^3 - \frac{3\,(rbig-rsmall)\ vol}{h\,\pi}\right)^{1/3}\right)}{rbig - rsmall}$

depthFromVolume2[generalFrustum, vol] → $\dfrac{h\ rbig - h^{2/3}\left(h\ rbig^3 + \frac{3\,(-rbig+rsmall)\ vol}{\pi}\right)^{1/3}}{rbig - rsmall}$

1

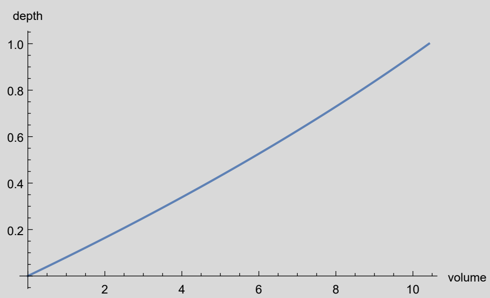
Testing

```
example = frustum[1, 2, π / 9, apexangle]
{ volume[example], volume[example] // N }
expr = test @ depthFromVolume[example, v];
Plot[expr, {v, 0, volume[example]}, AxesLabel → {"volume", "depth"}]
```

$\text{frustum}\left[1, 2, \frac{\pi}{9}, \text{apexangle}\right]$

$\left\{\frac{1}{3}\pi\left(12+\left(-6+\text{Tan}\left[\frac{\pi}{9}\right]\right)\text{Tan}\left[\frac{\pi}{9}\right]\right), 10.4182\right\}$

$\text{depthFromVolume}[\text{example}, v] \to \text{Cot}\left[\frac{\pi}{9}\right]\left(2-\left(8-\frac{3\,v\,\text{Tan}\left[\frac{\pi}{9}\right]}{\pi}\right)^{1/3}\right)$

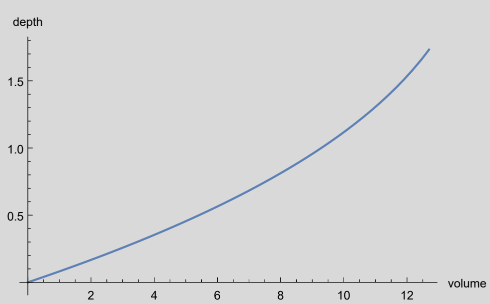



```
example = frustum[Sqrt[3], 2, 1]
{ volume[example], volume[example] // N }
expr = test @ depthFromVolume[example, v];
Plot[expr, {v, 0, volume[example]}, AxesLabel → {"volume", "depth"}]
```

$\text{frustum}\left[\sqrt{3}, 2, 1\right]$

$\left\{\frac{7\pi}{\sqrt{3}}, 12.6966\right\}$

$\text{depthFromVolume}[\text{example}, v] \to 2\sqrt{3} - 3^{1/3}\left(8\sqrt{3} - \frac{3\,v}{\pi}\right)^{1/3}$

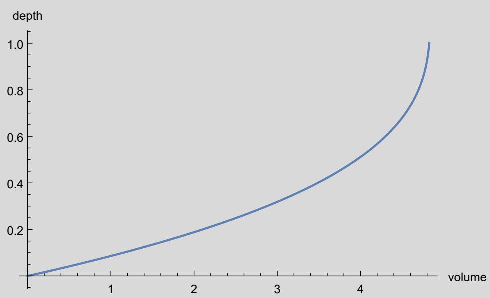


```
example = frustum[1, 2, π / 6, baseangle]
{ volume[example], volume[example] // N }
expr = test @ depthFromVolume[example, v];
Plot[expr, {v, 0, volume[example]}, AxesLabel → {"volume", "depth"}]
```

frustum$\left[1, 2, \frac{\pi}{6}, \text{baseangle}\right]$

$\left\{\left(5-2\sqrt{3}\right)\pi, 4.82517\right\}$

depthFromVolume[example, v] → $\dfrac{2-\left(8-\frac{3\sqrt{3}\,v}{\pi}\right)^{1/3}}{\sqrt{3}}$



Inverted Right Conical Frustum

Conversion

```
toFrustum[f : invertedFrustum[h_, rbig_, α_, apexangle]] := invert @ f
toFrustum[f : invertedFrustum[h_, rbig_, β_, baseangle]] := invert @ f
toFrustum[f : invertedFrustum[h_, rbig_, rsmall_]] := invert @ f

invert[f : frustum[h_, rbig_, α_, apexangle]] := invertedFrustum[h, rbig, α, apexangle]
invert[f : frustum[h_, rbig_, β_, baseangle]] := invertedFrustum[h, rbig, β, baseangle]
invert[f : frustum[h_, rbig_, rsmall_]] := invertedFrustum[h, rbig, rsmall]

invert[f : invertedFrustum[h_, rbig_, α_, apexangle]] := frustum[h, rbig, α, apexangle]
invert[f : invertedFrustum[h_, rbig_, β_, baseangle]] := frustum[h, rbig, β, baseangle]
invert[f : invertedFrustum[h_, rbig_, rsmall_]] := frustum[h, rbig, rsmall]
```

Accessing

```
assumptions[f : invertedFrustum[h_, rbig_, rsmall_]] := assumptions @ toFrustum @ f
assumptions[f : invertedFrustum[h_, rbig_, α_, apexangle]] := assumptions @ toFrustum @ f
assumptions[f : invertedFrustum[h_, rbig_, β_, baseangle]] := assumptions @ toFrustum @ f
test @ assumptions[invertedFrustum[h, rbig, α, apexangle]];
test @ assumptions[invertedFrustum[h, rbig, β, baseangle]];
```

assumptions[invertedFrustum[h, rbig, α, apexangle]] → h ≥ 0 && rbig ≥ 0 && 2 α < π && α > 0

assumptions[invertedFrustum[h, rbig, β, baseangle]] → h ≥ 0 && rbig ≥ 0 && β > 0 && 2 β < π

```
apexangle[f:invertedFrustum[h_, rbig_, α_, apexangle]] := apexangle @ invert @ f
apexangle[f:invertedFrustum[h_, rbig_, β_, baseangle]] := apexangle @ invert @ f
apexangle[f:invertedFrustum[h_, rbig_, rsmall_]] := apexangle @ invert @ f

baseangle[f:invertedFrustum[h_, rbig_, α_, apexangle]] := baseangle @ invert @ f
baseangle[f:invertedFrustum[h_, rbig_, β_, baseangle]] := baseangle @ invert @ f
baseangle[f: invertedFrustum[h_, rbig_, rsmall_]] := baseangle @ invert @ f

baseangle[f: invertedFrustum[h_, rbig_, rbig_ - h_ Cot[β_]]] := baseangle @ invert @ f

test @ apexangle[invertedFrustum[h, rbig, rsmall]];
test @ baseangle[invertedFrustum[h, rbig, rsmall]];
test @ { baseangle[invertedFrustum[1, 3, 2]], baseangle[invertedFrustum[Sqrt[3], 2, 1]]};
```

apexangle[invertedFrustum[h, rbig, rsmall]] → ArcTan[h, rbig - rsmall]

baseangle[invertedFrustum[h, rbig, rsmall]] → ArcTan[rbig - rsmall, h]

{baseangle[invertedFrustum[1, 3, 2]], baseangle[invertedFrustum[$\sqrt{3}$, 2, 1]]} → $\{\frac{\pi}{4}, \frac{\pi}{3}\}$

```
height[f:invertedFrustum[h_, rbig_, α_, apexangle]] := h
height[f:invertedFrustum[h_, rbig_, β_, baseangle]] := h
height[f:invertedFrustum[h_, rbig_, rsmall_]] := h
```

```
rbig[f:invertedFrustum[h_, rbig_, α_, apexangle]] := rbig
rbig[f:invertedFrustum[h_, rbig_, β_, baseangle]] := rbig
rbig[f:invertedFrustum[h_, rbig_, rsmall_]] := rbig
```

```
rsmall[f:invertedFrustum[h_, rbig_, α_, apexangle]] := rsmall @ invert @ f
rsmall[f:invertedFrustum[h_, rbig_, β_, baseangle]] := rsmall @ invert @ f
rsmall[f:invertedFrustum[h_, rbig_, rsmall_]] := rsmall
rsmall[f:invertedFrustum[h_, rbig_, ArcTan[rbig_ - rsmall_, h_], baseangle]] := rsmall
test @ rsmall[invertedFrustum[h, rbig, α, apexangle]];
test @ rsmall[invertedFrustum[h, rbig, β, baseangle]];
test @ rsmall[invertedFrustum[h, rbig, rsmall]];
```

rsmall[invertedFrustum[h, rbig, α, apexangle]] → rbig - h Tan[α]

rsmall[invertedFrustum[h, rbig, β, baseangle]] → rsmall - h Cot[β]

rsmall[invertedFrustum[h, rbig, rsmall]] → rsmall

Conversion Redux

```
toInvertedFrustum[f: invertedFrustum[h_, rbig_, α_, apexangle]] := invertedFrustum[h, rbig, rsmall[f]]
toInvertedFrustum[f: invertedFrustum[h_, rbig_, β_, baseangle]] := invertedFrustum[h, rbig, rsmall[f]]
toInvertedFrustum[f: invertedFrustum[h_, rbig_, rsmall_]] := f

toCartesian[f: invertedFrustum[h_, rbig_, α_, apexangle]] := toInvertedFrustum @ f
toCartesian[f: invertedFrustum[h_, rbig_, β_, baseangle]] := toInvertedFrustum @ f
toCartesian[f: invertedFrustum[h_, rbig_, rsmall_]] := toInvertedFrustum @ f

toApexAngled[f:invertedFrustum[h_, rbig_, α_, apexangle]] := f
toApexAngled[f:invertedFrustum[h_, rbig_, β_, baseangle]] := invert @ toApexAngled @ invert @ f
toApexAngled[f:invertedFrustum[h_, rbig_, rsmall_]] := invert @ toApexAngled @ invert @ f

toBaseAngled[f:invertedFrustum[h_, rbig_, α_, apexangle]] := invert @ toBaseAngled @ invert @ f
toBaseAngled[f:invertedFrustum[h_, rbig_, β_, baseangle]] := f
toBaseAngled[f:invertedFrustum[h_, rbig_, rsmall_]] := invert @ toBaseAngled @ invert @ f
```

```
test @ toCartesian @ invertedFrustum[h, rbig, β, baseangle];
test @ toBaseAngled @ %;
test @ toApexAngled @ %%;
test @ toFrustum @ %;
test @ toBaseAngled @ %%;
```

```
toCartesian[invertedFrustum[h, rbig, β, baseangle]] → invertedFrustum[h, rbig, rsmall - h Cot[β]]
```

```
toBaseAngled[%] → invertedFrustum[h, rbig, ArcTan[rbig - rsmall + h Cot[β], h], baseangle]
```

```
toApexAngled[%%] → invertedFrustum[h, rbig, ArcTan[h, rbig - rsmall + h Cot[β]], apexangle]
```

```
toFrustum[%] → frustum[h, rbig, ArcTan[h, rbig - rsmall + h Cot[β]], apexangle]
```

toBaseAngled[%%] → invertedFrustum$\left[\text{h, rbig, } \frac{\pi}{2} - \text{ArcTan[h, rbig} - \text{rsmall} + \text{h Cot}[\beta]\text{], baseangle}\right]$

```
test @ toBaseAngled @ invertedFrustum[h, rbig, rsmall];
test @ toCartesian @ %;
```

```
toBaseAngled[invertedFrustum[h, rbig, rsmall]] → invertedFrustum[h, rbig, ArcTan[rbig - rsmall, h], baseangle]
```

```
toCartesian[%] → invertedFrustum[h, rbig, rsmall]
```

Volume

```
coneHeight[f : invertedFrustum[h_, rbig_, α_, apexangle]] := coneHeight @ invert @ f
coneHeight[f : invertedFrustum[h_, rbig_, β_, baseangle]] := coneHeight @ invert @ f
coneHeight[f : invertedFrustum[h_, rbig_, rsmall_]] := coneHeight @ invert @ f

test @ coneHeight[invertedFrustum[h, rbig, α, apexangle]];
test @ coneHeight[invertedFrustum[h, rbig, β, baseangle]];
test @ toApexAngled @ invertedFrustum[h, rbig, β, baseangle];
test @ coneHeight @ %;
test @ coneHeight[invertedFrustum[h, rbig, rsmall]];
test @ coneHeight[invertedFrustum[1, 3, 2]];
```

```
coneHeight[invertedFrustum[h, rbig, α, apexangle]] → rbig Cot[α]
```

```
coneHeight[invertedFrustum[h, rbig, β, baseangle]] → rbig Tan[β]
```

toApexAngled[invertedFrustum[h, rbig, β, baseangle]] → invertedFrustum$\left[\text{h, rbig, } \frac{\pi}{2} - \beta\text{, apexangle}\right]$

```
coneHeight[%] → rbig Tan[β]
```

coneHeight[invertedFrustum[h, rbig, rsmall]] → $\frac{\text{h rbig}}{\text{rbig} - \text{rsmall}}$

```
coneHeight[invertedFrustum[1, 3, 2]] → 3
```

```
volume[f : invertedFrustum[h_, rbig_, rsmall_]] := volume @ invert @ f
volume[f : invertedFrustum[h_, rbig_, α_, apexangle]] := volume @ invert @ f
volume[f : invertedFrustum[h_, rbig_, β_, baseangle]] := volume @ invert @ f

v = test @ volume[invertedFrustum[h, r1, r2]]; (* compare to textbook answer 1/3 h π (r1^2+r1 r2+r2^2) *)
vα = test @ volume[invertedFrustum[h, r, α, apexangle]];
test @ toCartesian @ invertedFrustum[h, r, α, apexangle];
vα2 = test @ volume[%];
vβ = test @ volume[invertedFrustum[h, r, β, baseangle]];
test @ (v /. r2 → 0);
Clear[v, vα, vα2, vβ]
```

volume[invertedFrustum[h, r1, r2]] → $\frac{1}{3} h \pi (r1^2 + r1\, r2 + r2^2)$

volume[invertedFrustum[h, r, α, apexangle]] → $\frac{1}{3} h \pi (3 r^2 + h \operatorname{Tan}[\alpha] (-3 r + h \operatorname{Tan}[\alpha]))$

toCartesian[invertedFrustum[h, r, α, apexangle]] → invertedFrustum[h, r, r − h Tan[α]]

volume[%] → $\frac{1}{3} \pi \operatorname{Cot}[\alpha] (r^3 - (r - h \operatorname{Tan}[\alpha])^3)$

volume[invertedFrustum[h, r, β, baseangle]] → $\frac{1}{3} h \pi (3 r^2 + h \operatorname{Cot}[\beta] (-3 r + h \operatorname{Cot}[\beta]))$

(v /. r2 → 0) → $\frac{1}{3} h \pi\, r1^2$

Height and Depth

Final

We’re looking for a frustum with same base angle and bottom radius, but different height

```
depthFromVolume[f : invertedFrustum[h_, rbig_, α_, apexangle], vol_] := Module[{},
  h - depthFromVolume[invert @ f, volume[f] - vol] // FullSimplify
 ]
generalApexInvertedFrustum = invertedFrustum[h, r, α, apexangle]
test @ depthFromVolume[generalApexInvertedFrustum, vol];
```

invertedFrustum[h, r, α, apexangle]

depthFromVolume[generalApexInvertedFrustum, vol] → $h + \operatorname{Cot}[\alpha] \left(-r + \left(r^3 + \frac{\operatorname{Tan}[\alpha] (-3 h \pi r^2 + 3\, vol + h^2 \pi \operatorname{Tan}[\alpha] (3 r - h \operatorname{Tan}[\alpha]))}{\pi}\right)^{1/3}\right)$

```
depthFromVolume[f : invertedFrustum[h_, rbig_, rsmall_], vol_] := Module[{},
  h - depthFromVolume[invert @ f, volume[f] - vol] // FullSimplify
 ]
generalInvertedFrustum = invertedFrustum[h, rbig, rsmall]
test @ depthFromVolume[generalInvertedFrustum, vol];
```

invertedFrustum[h, rbig, rsmall]

depthFromVolume[generalInvertedFrustum, vol] → $\frac{h\, rsmall - h^{2/3} \left(h\, rsmall^3 + \frac{3 (rbig - rsmall)\, vol}{\pi}\right)^{1/3}}{-rbig + rsmall}$

```
depthFromVolume[f : invertedFrustum[h_, rbig_, β_, baseangle], vol_] := Module[{hh, rr, αα, vv, eqn, soln},
  h - depthFromVolume[invert @ f, volume[f] - vol] // FullSimplify
 ]
generalBaseInvertedFrustum = invertedFrustum[h, r, β, baseangle]
test @ depthFromVolume[generalBaseInvertedFrustum, vol];
```

```
invertedFrustum[h, r, β, baseangle]
```

depthFromVolume[generalBaseInvertedFrustum, vol] $\to h + \left(-r + \left(r^3 + \frac{\mathrm{Cot}[\beta]\,\left(-3\,h\,\pi\,r^2 + 3\,\mathrm{vol} + h^2\,\pi\,\mathrm{Cot}[\beta]\,(3\,r - h\,\mathrm{Cot}[\beta])\right)}{\pi}\right)^{1/3}\right)\mathrm{Tan}[\beta]$

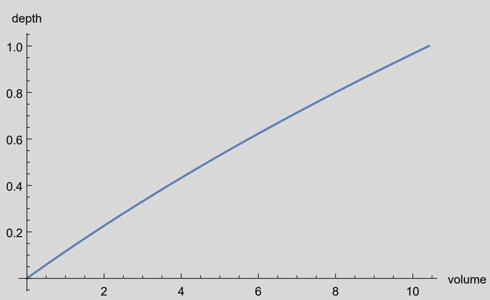
Testing

```
example = invertedFrustum[1, 2, π / 9, apexangle]
{ volume[example], volume[example] // N }
expr = test @ depthFromVolume[example, v];
Plot[expr, {v, 0, volume[example]}, AxesLabel → {"volume", "depth"}]
```

invertedFrustum$\left[1, 2, \frac{\pi}{9}, \mathrm{apexangle}\right]$

$\left\{\frac{1}{3}\pi\left(12 + \left(-6 + \mathrm{Tan}\left[\frac{\pi}{9}\right]\right)\mathrm{Tan}\left[\frac{\pi}{9}\right]\right), 10.4182\right\}$

depthFromVolume[example, v] $\to 1 - 2\,\mathrm{Cot}\left[\frac{\pi}{9}\right] + \frac{\left(3\,v\,\mathrm{Cot}\left[\frac{\pi}{9}\right]^2 + \pi\left(-1 + 2\,\mathrm{Cot}\left[\frac{\pi}{9}\right]\right)^3\right)^{1/3}}{\pi^{1/3}}$

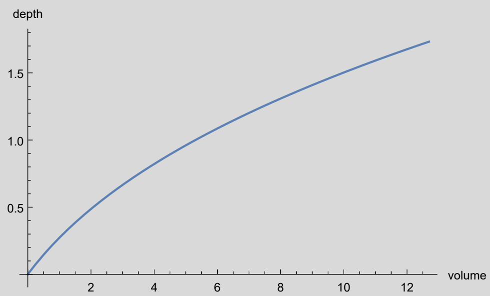


```
example = invertedFrustum[Sqrt[3], 2, 1]
{ volume[example], volume[example] // N }
expr = test @ depthFromVolume[example, v];
Plot[expr, {v, 0, volume[example]}, AxesLabel → {"volume", "depth"}]
```

$\text{invertedFrustum}\left[\sqrt{3}, 2, 1\right]$

$\left\{\frac{7\pi}{\sqrt{3}}, 12.6966\right\}$

$\text{depthFromVolume}[\text{example}, \text{v}] \to -\sqrt{3} + \left(3\sqrt{3} + \frac{9\,\text{v}}{\pi}\right)^{1/3}$

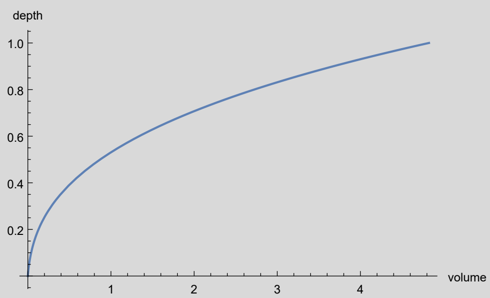



```
example = invertedFrustum[1, 2, π / 6, baseangle]
{ volume[example], volume[example] // N }
expr = test @ depthFromVolume[example, v];
Plot[expr, {v, 0, volume[example]}, AxesLabel → {"volume", "depth"}]
```

$\text{invertedFrustum}\left[1, 2, \frac{\pi}{6}, \text{baseangle}\right]$

$\left\{\left(5 - 2\sqrt{3}\right)\pi, 4.82517\right\}$

$\text{depthFromVolume}[\text{example}, \text{v}] \to 1 - \frac{2}{\sqrt{3}} + \frac{\left(26 - 15\sqrt{3} + \frac{3\sqrt{3}\,\text{v}}{\pi}\right)^{1/3}}{\sqrt{3}}$



Sphere

Accessing

```
assumptions[sphere[r_]] := r ≥ 0
radius[sphere[r_]] := r
```

Volume

```
volume[sphere[r_]] := Module[{α},
  4 / 3 Pi r^3
 ]
test @ volume[sphere[r]];
```

$$\text{volume[sphere[r]]} \to \frac{4\pi r^3}{3}$$

Inverted Spherical Cap

See http://mathworld.wolfram.com/SphericalCap.html. By ‘inverted’ spherical cap, we mean a cap on the bottom of the sphere instead of the top.

Accessing

```
Solve[r - h == r Sin[α], h]
```

```
{{h → r - r Sin[α]}}
```

```
assumptions[invertedSphericalCap[r_, h_]] := r > 0 && h > 0 && r ≥ h
assumptions[invertedSphericalCap[r_, α_, angled]] := r > 0 && α ≥ 0 && α < π / 2

radius[c : invertedSphericalCap[r_, h_]] := r
height[c : invertedSphericalCap[r_, h_]] := h
angle[invertedSphericalCap[r_, h_]] := ArcSin[(r - h) / r]

radius[c : invertedSphericalCap[r_, α_, angled]] := r
height[c : invertedSphericalCap[r_, α_, angled]] := r - r Sin[α]
angle[invertedSphericalCap[r_, α_, angled]] := α
```

Conversion

```
toCartesian[c : invertedSphericalCap[r_, h_]] := c
toAngled[c : invertedSphericalCap[r_, h_]] := invertedSphericalCap[r, angle[c], angle]

toCartesian[c : invertedSphericalCap[r_, α_, angled]] := invertedSphericalCap[r, height[c]]
toAngled[c : invertedSphericalCap[r_, α_, angled]] := c

test @ toCartesian @ invertedSphericalCap[r, α, angled];
test @ toAngled @ toCartesian @ invertedSphericalCap[r, α, angled];
```

```
toCartesian[invertedSphericalCap[r, α, angled]] → invertedSphericalCap[r, r - r Sin[α]]
```

```
toAngled[toCartesian[invertedSphericalCap[r, α, angled]]] → invertedSphericalCap[r, ArcSin[Sin[α]], angle]
```

Volume

```
volume[invertedSphericalCap[r_, h_]] := Module[{},
  (* http://mathworld.wolfram.com/SphericalCap.html *)
  π / 3 * h^2 * (3 r - h)
 ]
volume[invertedSphericalCap[r_, α_, angled]] := Module[{},
  π / 3 r^3 (2 - 3 Sin[α] + Sin[α]^3)
 ]
test @ volume[invertedSphericalCap[r, h]];
test @ volume[invertedSphericalCap[r, α, angled]];
```

volume[invertedSphericalCap[r, h]] → $\frac{1}{3} h^2 \pi (-h + 3 r)$

volume[invertedSphericalCap[r, α, angled]] → $\frac{1}{3} \pi r^3 \left(2 - 3 \,\text{Sin}[\alpha] + \text{Sin}[\alpha]^3\right)$

Height and Depth

Final

```
genericSphericalCapDepthFromVolumeCartesian[] := Module[{cap, r, h, vol, a, eqn, solns, soln},
  cap = invertedSphericalCap[r, h];
  a = assumptions[cap] && vol ≥ 0;
  eqn = vol == volume[cap];
  solns = Assuming[a, Solve[eqn, h]];
  soln = h /. solns[[3]];
  genericSphericalCapDepthFromVolumeCartesian[] = {h, r , vol, soln}
 ]
test @ genericSphericalCapDepthFromVolumeCartesian[];
```

genericSphericalCapDepthFromVolumeCartesian[] →

$$\left\{\text{h\$28051}, \text{r\$28051}, \text{vol\$28051}, \text{r\$28051} - \frac{\left(1 - i \sqrt{3}\right) \pi^{1/3}\, \text{r\$28051}^2}{2^{2/3} \left(2 \pi\, \text{r\$28051}^3 - 3\, \text{vol\$28051} + \sqrt{3} \sqrt{-4 \pi\, \text{r\$28051}^3\, \text{vol\$28051} + 3\, \text{vol\$28051}^2}\right)^{1/3}} - \frac{\left(1 + i \sqrt{3}\right) \left(2 \pi\, \text{r\$28051}^3 - 3\, \text{vol\$28051} + \sqrt{3} \sqrt{-4 \pi\, \text{r\$28051}^3\, \text{vol\$28051} + 3\, \text{vol\$28051}^2}\right)^{1/3}}{2\, (2 \pi)^{1/3}}\right\}$$

```
(* not used *)
genericSphericalCapDepthFromVolumeAngled[] := Module[{cap, r, α, vol, a, eqn, solns, soln},
  cap = invertedSphericalCap[r, α, angled];
  a = assumptions[cap] && vol ≥ 0;
  eqn = vol == volume[cap];
  solns = Assuming[a, Solve[eqn, α]];
  ((α /. # /. C[1] → 0) & /@solns) [[{4, 6}]] (* 4 & 6 are empirical*)
 ]
test @ genericSphericalCapDepthFromVolumeAngled[];
```

genericSphericalCapDepthFromVolumeAngled[] →

$$\left\{\text{ArcSin}\left[\frac{\left(1+i\sqrt{3}\right)\pi^{1/3}\,\text{r\$28060}^3}{2^{2/3}\left(2\pi\,\text{r\$28060}^9-3\,\text{r\$28060}^6\,\text{vol\$28060}+\sqrt{3}\sqrt{-4\pi\,\text{r\$28060}^{15}\,\text{vol\$28060}+3\,\text{r\$28060}^{12}\,\text{vol\$28060}^2}\right)^{1/3}}+\frac{\left(1-i\sqrt{3}\right)\left(2\pi\,\text{r\$28060}^9-3\,\text{r\$28060}^6\,\text{vol\$28060}+\sqrt{3}\sqrt{-4\pi\,\text{r\$28060}^{15}\,\text{vol\$28060}+3\,\text{r\$28060}^{12}\,\text{vol\$28060}^2}\right)^{1/3}}{2\,(2\pi)^{1/3}\,\text{r\$28060}^3}\right],\right.$$
$$\left.\text{ArcSin}\left[\frac{\left(1-i\sqrt{3}\right)\pi^{1/3}\,\text{r\$28060}^3}{2^{2/3}\left(2\pi\,\text{r\$28060}^9-3\,\text{r\$28060}^6\,\text{vol\$28060}+\sqrt{3}\sqrt{-4\pi\,\text{r\$28060}^{15}\,\text{vol\$28060}+3\,\text{r\$28060}^{12}\,\text{vol\$28060}^2}\right)^{1/3}}+\frac{\left(1+i\sqrt{3}\right)\left(2\pi\,\text{r\$28060}^9-3\,\text{r\$28060}^6\,\text{vol\$28060}+\sqrt{3}\sqrt{-4\pi\,\text{r\$28060}^{15}\,\text{vol\$28060}+3\,\text{r\$28060}^{12}\,\text{vol\$28060}^2}\right)^{1/3}}{2\,(2\pi)^{1/3}\,\text{r\$28060}^3}\right]\right\}$$

```
depthFromVolume[c : invertedSphericalCap[r_, α_, angled], v_] := depthFromVolume[toCartesian @ c, v]
depthFromVolume[c : invertedSphericalCap[r_, h_], v_] := Module[{rr, hh, vol, soln},
  assert[assumptions[c]];
  {hh, rr, vol, soln} = genericSphericalCapDepthFromVolumeCartesian[];
  (soln /. {rr → r, hh → h, vol → v})
 ]
test @ depthFromVolume[invertedSphericalCap[2, 1], volume];
% /. volume → 1 // N
test @ depthFromVolume[invertedSphericalCap[r, h], volume];
N @ %
```

depthFromVolume[invertedSphericalCap[2, 1], volume] →

$$2-\frac{2\left(1-i\sqrt{3}\right)(2\pi)^{1/3}}{\left(16\pi-3\,\text{volume}+\sqrt{3}\sqrt{-32\pi\,\text{volume}+3\,\text{volume}^2}\right)^{1/3}}-\frac{\left(1+i\sqrt{3}\right)\left(16\pi-3\,\text{volume}+\sqrt{3}\sqrt{-32\pi\,\text{volume}+3\,\text{volume}^2}\right)^{1/3}}{2\,(2\pi)^{1/3}}$$

$0.413441+4.44089\times10^{-16}\,i$

depthFromVolume[invertedSphericalCap[r, h], volume] →

$$r-\frac{\left(1-i\sqrt{3}\right)\pi^{1/3}\,r^2}{2^{2/3}\left(2\pi r^3-3\,\text{volume}+\sqrt{3}\sqrt{-4\pi r^3\,\text{volume}+3\,\text{volume}^2}\right)^{1/3}}-\frac{\left(1+i\sqrt{3}\right)\left(2\pi r^3-3\,\text{volume}+\sqrt{3}\sqrt{-4\pi r^3\,\text{volume}+3\,\text{volume}^2}\right)^{1/3}}{2\,(2\pi)^{1/3}}$$

$$r-\frac{(0.922635-1.59805\,i)\,r^2}{\left(6.28319\,r^3-3.\,\text{volume}+1.73205\sqrt{-12.5664\,r^3\,\text{volume}+3.\,\text{volume}^2}\right)^{1/3}}-(0.270963+0.469322\,i)\left(6.28319\,r^3-3.\,\text{volume}+1.73205\sqrt{-12.5664\,r^3\,\text{volume}+3.\,\text{volume}^2}\right)^{1/3}$$

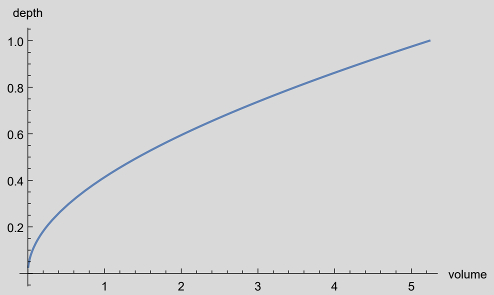
Testing

```
example = invertedSphericalCap[2, 1]
{ volume[example], volume[example] // N }
expr = test @ depthFromVolume[example, v];
Plot[expr, {v, 0, volume[example]}, AxesLabel → {"volume", "depth"}]
```

```
invertedSphericalCap[2, 1]
```

$$\left\{\frac{5\pi}{3}, 5.23599\right\}$$

$$\text{depthFromVolume[example, v]} \to 2 - \frac{2\left(1 - i\sqrt{3}\right)(2\pi)^{1/3}}{\left(16\pi - 3v + \sqrt{3}\sqrt{-32\pi v + 3v^2}\right)^{1/3}} - \frac{\left(1 + i\sqrt{3}\right)\left(16\pi - 3v + \sqrt{3}\sqrt{-32\pi v + 3v^2}\right)^{1/3}}{2(2\pi)^{1/3}}$$



Unknown Shape

Accessing

```
assumptions[u : unknownShape[h_, vol_]] := h ≥ 0 && vol ≥ 0
test @ assumptions[unknownShape[h, vol]];
```

```
assumptions[unknownShape[h, vol]] → h ≥ 0&&vol ≥ 0
```

```
height[u : unknownShape[h_, vol_]] := h
toCartesian[u : unknownShape[h_, vol_]] := u
```

```
volume[u : unknownShape[h_, vol_]] := Module[{},
  (*printCell[{volume, "h" → h, "vol" → vol}];*)
  vol]
```

```
depthFromVolume[u : unknownShape[h_, vol_], v_] := Module[{},
  (*printCell[{depthFromVolume, "h" → h, "vol" → vol, "v" → v}];*)
  If[v ≤ 0 || h ≤ 0 || vol ≤ 0,
   0,
   Indeterminate]]
```

Conical Test Tube

Our model of a conical test tube is an “cylindrical” inverted frustum on top of a “conical” inverted frustum on top of an inverted spherical cap

Accessing

```
assumptions[conicalTestTube[cylindrical_, conical_, cap_]] :=
 assumptions[cylindrical] && assumptions[conical] && assumptions[cap]
```

```
toCanonical[c : conicalTestTube[cylindrical_, conical_, cap_]] := c
toCanonical[conicalTestTube[{idTop_, idHip_, idBottom_}, {hTop_, hBottomAndCap_}]] := conicalTestTube[
  (* TODO: use cylinders when we need to *)
  invertedFrustum[hTop, idTop / 2 , idHip / 2 ],
  invertedFrustum[hBottomAndCap - idBottom, idHip / 2 , idBottom / 2 ],
  invertedSphericalCap[idBottom / 2, idBottom / 2 ]
 ]
```

```
toCartesian[c : conicalTestTube[cylindrical_, conical_, cap_]] := Map[toCartesian, c, {1}]
toApexAngled[c : conicalTestTube[cylindrical_, conical_, cap_]] := Map[toApexAngled, c, {1}]
toBaseAngled[c : conicalTestTube[cylindrical_, conical_, cap_]] := Map[toBaseAngled, c, {1}]
test @ toCartesian[conicalTestTube[cylindrical, conical, cap]];
```

```
toCartesian[conicalTestTube[cylindrical, conical, cap]] →
 conicalTestTube[toCartesian[cylindrical], toCartesian[conical], toCartesian[cap]]
```

```
height[c : conicalTestTube[cylindrical_, conical_, cap_]] := Total @ (List @@ Map[height, c, {1}])
```

```
parts[c : conicalTestTube[cylindrical_, conical_, cap_]] :=
 {"cylindrical" → cylindrical, "conical" → conical, "cap" → cap} // Association
parts[c : conicalTestTube[idTop_, idHip_, idBottom_, hTop_, hBottom_]] := parts @ toCanonical @ c
test @ parts[toCanonical @ conicalTestTube[{idTop, idHip, idBottom}, {hTop, hBottom}]];
```

parts[toCanonical[conicalTestTube[{idTop, idHip, idBottom}, {hTop, hBottom}]]] →

$\langle|$ cylindrical → invertedFrustum$\left[\text{hTop}, \frac{\text{idTop}}{2}, \frac{\text{idHip}}{2}\right]$,
conical → invertedFrustum$\left[\text{hBottom} - \text{idBottom}, \frac{\text{idHip}}{2}, \frac{\text{idBottom}}{2}\right]$, cap → invertedSphericalCap$\left[\frac{\text{idBottom}}{2}, \frac{\text{idBottom}}{2}\right]$ $|\rangle$

Volume

```
volume[c : conicalTestTube[cylindrical_, conical_, cap_]] := Total[volume /@ parts[c]]
volume[c : conicalTestTube[idTop_, idHip_, idBottom_, hTop_, hBottom_]] := volume @ toCanonical @ c
```

Height & Depth

Math

```
depthFromVolume[c : conicalTestTube[{idTop_, idHip_, idBottom_}, {hTop_, hBottom_}], v_] := depthFromVolume[toCanonical @ c, v]
depthFromVolume[c : conicalTestTube[cylindrical_, conical_, cap_], v_] :=
 Module[{vCylindrical, vConical, vCap, dFromCap, dFromConical, dOther, result},
  vCap = volume[cap];
  vConical = volume[conical];
  dFromCap = depthFromVolume[cap, v];
  dFromConical = height[cap] + depthFromVolume[conical, v - vCap];
  dOther = height[cap] + height[conical] + depthFromVolume[cylindrical, v - vCap - vConical];
  Piecewise[
   {
    {dFromCap, v ≤ vCap},
    {dFromConical, v ≤ vConical},
    {dOther, True}
   }
  ]
 ]
```

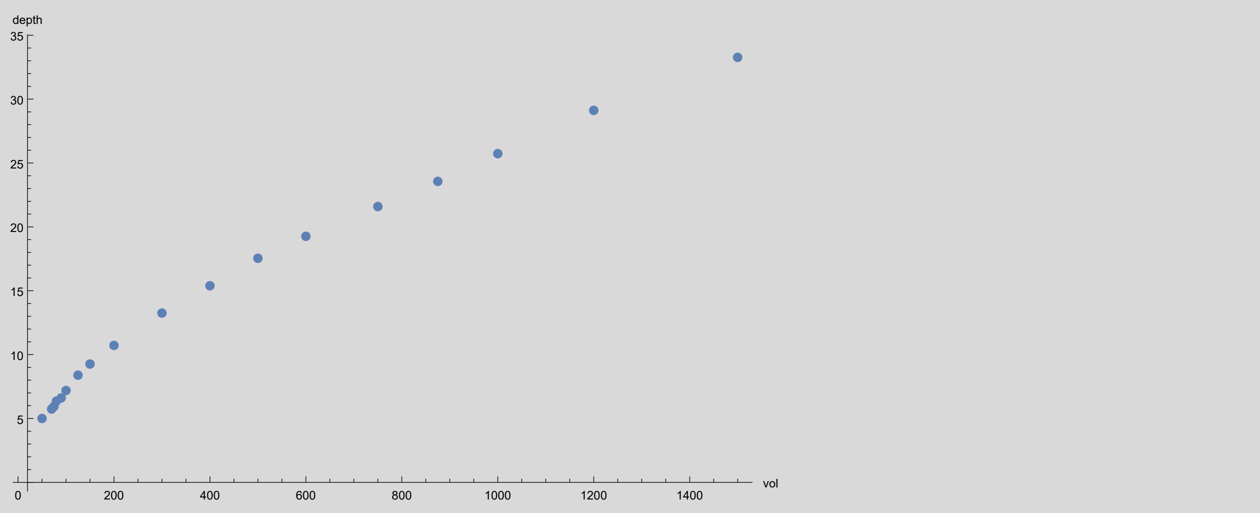
Examples

Eppendorf Tubes

Data

```
eppendorfData = ArrayReshape[{50, 5, 70, 5.74, 75, 5.94, 80, 6.36, 90, 6.61, 100, 7.19, 125, 8.39, 150, 9.26, 200, 10.72, 300,
   13.25, 400, 15.39, 500, 17.54, 600, 19.26, 750, 21.59, 875, 23.56, 1000, 25.73, 1200, 29.12, 1500, 33.27}, {18, 2}]
ListPlot[eppendorfData, ImageSize → Large, AxesLabel → {"vol", "depth"}, PlotRange → All]
```

```
{{50, 5}, {70, 5.74}, {75, 5.94}, {80, 6.36}, {90, 6.61}, {100, 7.19}, {125, 8.39}, {150, 9.26}, {200, 10.72}, {300, 13.25},
 {400, 15.39}, {500, 17.54}, {600, 19.26}, {750, 21.59}, {875, 23.56}, {1000, 25.73}, {1200, 29.12}, {1500, 33.27}}
```

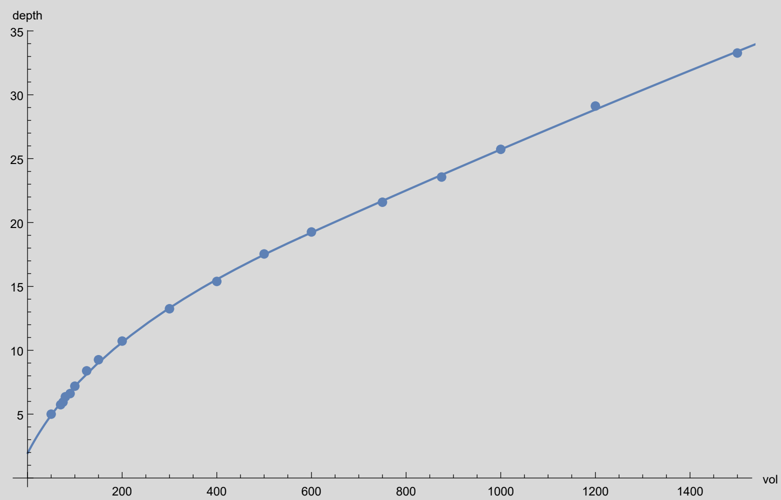


Fitting

```
fitEppendorfData[eppendorfData_] := Module[
  {depthFunc, fit, showFit, zeroify, conicalData, conePart, coneRules, angledCone, cylinderData, offsetConicalData,
   offsetCylinderData, cylinderPart, cylinderRules, hCone, hCyl, rtop, rmid, rbottom, angledCylinder, specRules, hTot,
   tube, α, tubeRules, rconeBig, rconeSmall, wallBottom, rules, αCylinder, αCone, hCap, rCap, volCap, fittedTube},
  depthFunc[part_] := Module[{expr, v},
    expr = depthFromVolume[part, v];
    depthFunc[part] = Function[{vol}, expr /. {v → vol}]];
  fit[part_, assump_, vars_, data_] := Module[{errors, err, min, fitRules, asses},
    errors = Function[{vol, depth},
         (depthFunc[part][vol] - depth)^2
       ] @@ # & /@ data;
    err = Total[errors] // N;
    asses = assumptions[part] && (And @@ assump);
    (*test @ asses;*)
    {min, fitRules} = NMinimize[{err, asses }, vars];
    fitRules];
  showFit[part_, data_] := Module[{v},
    Show[ListPlot[{data}, ImageSize → Large, AxesLabel → {"vol", "depth"}, PlotRange → All, AxesOrigin → {0, 0}],
     Plot[depthFromVolume[part, v], {v, 0, volume[part]}]]];
  zeroify[data_] := Module[{xMin, yMin},
    {xMin, yMin} = Map[Min, Transpose @ data, {1}];
    Transpose[Transpose[data] - {xMin, yMin}]];

  conicalData = Select[eppendorfData, #[[1]] ≤ 500 &];
  cylinderData = Select[eppendorfData, #[[1]] >= 500 &]; (* hard to tell for in between data, so we're conservative *)
  offsetConicalData = zeroify[conicalData];
```

```
    offsetCylinderData = zeroify[cylinderData];
    (*printCell @ ListPlot[{conicalData, cylinderData}, ImageSize→Large, AxesLabel→{"vol", "depth"}, PlotRange→All];*)
    (*printCell @ ListPlot[{offsetCylinderData}, ImageSize→Large, AxesLabel→{"vol", "depth"}, PlotRange→All];*)
    specRules = { hTot → 37.8, rmid → 8.7 / 2, wallBottom → 38.9 - 37.8};
    printCell[specificationSays[specRules]];

    (* fit the cylinder. this gives us the apex angle of the cylinder. we don't yet know its actual height *)
    (* we dont' know rmid because the bottom of cylinderData might not be right at the mid location *)
    cylinderPart = invertedFrustum[hCyl, rtop, rmid](* /. coneRules*);
    cylinderRules = fit[cylinderPart, {hCyl > 12}, {hCyl, rtop, rmid}, offsetCylinderData];
    angledCylinder = toApexAngled[cylinderPart /. cylinderRules];
    (*test @ cylinderRules;
    test @ (cylinderPart /. cylinderRules);
    test @ angledCylinder;
    test @ toDeg @ apexangle[angledCylinder];*)
    (*printCell @ showFit[cylinderPart /. cylinderRules, offsetCylinderData];*)

    (* fit the cone. this gives us the apex angle of the cone *)
    conePart = invertedFrustum[hCone, rconeBig, rconeSmall];
    coneRules = fit[conePart, {hCone > 10}, {hCone, rconeBig, rconeSmall}, offsetConicalData];
    angledCone = toApexAngled[conePart /. coneRules];
    (*test @ coneRules;
    test @ (conePart /. coneRules);
    test @ angledCone;
    test @ toDeg @ apexangle[angledCone];*)
    (*printCell @ showFit[conePart /. coneRules, offsetConicalData];*)

    (* summarize what we know *)
    rules = {αCylinder → apexangle[angledCylinder], αCone → apexangle[angledCone]};
    (*test @ rules;*)

    (* put these together. *)
    (* Cap is just a shape that can fix a volume; we have no data in that range, and can't measure volumes therein. *)
    tube = conicalTestTube[
      (invertedFrustum[hCyl, rbig[hCyl, rmid, αCylinder, apexangle], αCylinder, apexangle] /. rules),
      (invertedFrustum[hCone, rmid, αCone, apexangle] /. rules),
      (unknownShape[hCap, volCap])
     ];
    tube = tube /. { hCone → (hTot /. specRules) - hCyl - hCap};
    (*test @ tube;*)
    tubeRules =
     fit[tube, {hCap < 5, hCyl > 10, rmid > 4, rmid < 6(*, rCap ≥ hCap*)}, {hCyl, rmid, hCap, volCap}, eppendorfData];
    fittedTube = toCartesian[tube /. tubeRules];
    (*test @ tubeRules;
    test @ fittedTube;*)
    printCell @ showFit[fittedTube, eppendorfData];
    fittedTube
  ]
fittedEppendorf = fitEppendorfData[eppendorfData]
test @ height @ fittedEppendorf;
test @ depthFromVolume[fittedEppendorf, volume[fittedEppendorf]];
test @ volume @ fittedEppendorf;
```

```
specificationSays[{hTot$28213 → 37.8, rmid$28213 → 4.35, wallBottom$28213 → 1.1}]
```




```
conicalTestTube[invertedFrustum[18.9894, 4.70751, 4.35636],
 invertedFrustum[16.8419, 4.35636, 2.1099], unknownShape[1.96866, 0.550217]]
```

```
height[fittedEppendorf] → 37.8
```

```
depthFromVolume[fittedEppendorf, volume[fittedEppendorf]] → 37.8
```

```
volume[fittedEppendorf] → 1801.76
```

It’s regrettable that we don’t bottom out at 0 mm (we bottom out at about 2 mm), but the data does really fit quite nicely otherwise.

It should be noted that the specification indicates that the upper ‘cylindrical’ inverted frustum isn’t actually an inverted frustum but has a bit of a flare at the top.

Bio-rad Deep Well Plates

The Bio-rad specs aren’t internally consistent: there’s a conflict between the well diameters and height vs the well angle.

We first choose to honor the well bottom width (2.64).

```
modelBioRad1[] := Module[{cylinderPart, cylinderRules, capacity, hCyl, rtop, rmid, rbottom, conePart, hCone,
    specRules, rules, hTot, wallBottom, αCone, tube, vol, solns, soln, assumpts, constraint, extra, hCylMin },

   (* we assume the top is an actual cylinder rather than an inverted frustum *)
   cylinderPart = cylinder[hCyl, rtop];
   cylinderRules = {rmid → rtop};

   conePart = invertedFrustum[hCone, rmid, αCone, apexangle]; (* doesn't honor rbottom on its own *)
   conePart = invertedFrustum[hCone, rmid, rbottom];

   specRules = { hTot → 14.81, rtop → 5.46 / 2, rbottom → 2.64 / 2, wallBottom → 16.06 - 14.81, αCone → toRadian[17.5] / 2};
   (*printCell[specificationSays[specRules]];*)
   tube = conicalTestTube[cylinderPart, conePart, emptyCylinder[]];
   rules = {hCone → hTot - hCyl } ~Join~ cylinderRules ~Join~ specRules;
   tube = tube //. rules;
   vol = volume[tube];
   capacity = 200 ;
   assumpts =  True;
   solns = Solve[vol == capacity && assumpts, {hCyl}];
   soln = First @ solns;
   tube //. soln // toCartesian
  ]
modelledBioRad1 = modelBioRad1[];
test @ modelledBioRad1;
test @ toDeg[apexangle[parts[modelledBioRad1]["conical"]] * 2];
test @ (2 * rsmall[parts[modelledBioRad1]["conical"]]);
```

```
modelledBioRad1 → conicalTestTube[cylinder[0.150026, 2.73], invertedFrustum[14.66, 2.73, 1.32], cylinder[0, 0]]
```

```
toDeg[apexangle[parts[modelledBioRad1][conical]] 2] → 10.9876
```

```
2 rsmall[parts[modelledBioRad1][conical]] → 2.64
```

So instead we honor the apex angle of the cone (17.5°).

```
modelBioRad2[] := Module[{cylinderPart, cylinderRules, capacity, hCyl, rtop, rmid, rbottom, conePart, hCone,
    specRules, rules, hTot, wallBottom, αCone, tube, vol, solns, soln, assumpts, constraint, extra, hCylMin },

  (* we assume the top is an actual cylinder rather than an inverted frustum *)
  cylinderPart = cylinder[hCyl, rtop];
  cylinderRules = {rmid → rtop};

  conePart = invertedFrustum[hCone, rmid, αCone, apexangle]; (* doesn't honor rbottom on its own *)

  specRules = { hTot → 14.81, rtop → 5.46 / 2, rbottom → 2.64 / 2, wallBottom → 16.06 - 14.81, αCone → toRadian[17.5] / 2};
  (*printCell[specificationSays[specRules]];*)
  tube = conicalTestTube[cylinderPart, conePart, emptyCylinder[]];
  rules = {hCone → hTot - hCyl } ~ Join ~ cylinderRules ~ Join ~ specRules;
  tube = tube //. rules;
  vol = volume[tube];
  capacity = 200 ;
  assumpts = hCyl > 0 && hCyl < 5;
  solns = Solve[vol == capacity && assumpts, {hCyl}];
  soln = First @ solns;
  tube //. soln // toCartesian
 ]
modelledBioRad2 = modelBioRad2[];
test @ modelledBioRad2;
test @ toDeg[apexangle[parts[modelledBioRad2]["conical"]] * 2];
test @ (2 * rsmall[parts[modelledBioRad2]["conical"]]);
```

```
modelledBioRad2 → conicalTestTube[cylinder[2.83192, 2.73], invertedFrustum[11.9781, 2.73, 0.886397], cylinder[0, 0]]
```

```
toDeg[apexangle[parts[modelledBioRad2][conical]] 2] → 17.5
```

```
2 rsmall[parts[modelledBioRad2][conical]] → 1.77279
```

Next, we honor *both* the apex angle and the bottom dimension. But to do that, we need to admit that the capacity of the well is greater than stated (which is almost certainly true).

```
modelBioRad3[] := Module[{cylinderPart, cylinderRules, capacity, hCyl, rtop, rmid, rbottom, conePart, hCone, specRules,
   rules, hTot, wallBottom, αCone, tube, vol, solns, soln, assumpts, constraint, extra, hCylMin, hCylSoln },

  (* we assume the top is an actual cylinder rather than an inverted frustum *)
  cylinderPart = cylinder[hCyl, rtop];
  cylinderRules = {rmid → rtop};

  conePart = invertedFrustum[hCone, rmid, αCone, apexangle]; (* doesn't honor rbottom on its own *)

  specRules = { hTot → 14.81, rtop → 5.46 / 2, rbottom → 2.64 / 2, wallBottom → 16.06 - 14.81, αCone → toRadian[17.5] / 2};
  (*printCell[specificationSays[specRules]];*)
  tube = conicalTestTube[cylinderPart, conePart, emptyCylinder[]];
  rules = {hCone → hTot - hCyl } ~Join~ cylinderRules ~Join~ specRules;
  tube = tube //. rules;

  constraint = (rsmall[conePart] - rbottom) //. rules;
  hCylSoln = First @ Solve[constraint == 0, {hCyl}];
  tube = tube //. hCylSoln;

  vol = volume[tube];
  capacity = 200 + extra;
  assumpts = extra ≥ 0;
  solns = Solve[vol == capacity && assumpts, {extra}];
  soln = First @ solns;
  tube //. soln // toCartesian
 ]
modelledBioRad3 = modelBioRad3[];
test @ modelledBioRad3;
test @ toDeg[apexangle[parts[modelledBioRad3]["conical"]] * 2];
test @ (2 * rsmall[parts[modelledBioRad3]["conical"]]);
test @ volume[modelledBioRad3];
```

```
modelledBioRad3 → conicalTestTube[cylinder[5.64908, 2.73], invertedFrustum[9.16092, 2.73, 1.32], cylinder[0, 0]]
```

```
toDeg[apexangle[parts[modelledBioRad3][conical]] 2] → 17.5
```

```
2 rsmall[parts[modelledBioRad3][conical]] → 2.64
```

```
volume[modelledBioRad3] → 255.051
```

Previous Work

```
example = Module[{cone, α, rsmall, rbig, hOverall, h},
  α = toRadian[17.5] / 2;
  rsmall = 2.64 / 2;
  rbig = 5.46 / 2;
  hOverall = 14.81;
  h = 14.66; (* from a previous call to Solve *)
  conicalTestTube[cylinder[hOverall - h, rbig], invertedFrustum[h, rbig, rsmall], emptyCylinder[]]]
volume @ example
Solve[% == 200, h]
```

```
conicalTestTube[cylinder[0.15, 2.73], invertedFrustum[14.66, 2.73, 1.32], cylinder[0, 0]]
```

```
200.
```

```
{}
```

If we honor the well angle, then the well diameter at opening is too small. Maybe we can’t ignore the cap?

```
example = Module[{f},
  f = invertedFrustum[h, rbig, toRadian[17.5] / 2, apexangle];
  conicalTestTube[
   cylinder[14.81 - h, rbig],
   f,
   emptyCylinder[]]]
volume @ example == 200
rsmall[parts[example]["conical"]] == 2.64 / 2
Solve[{%%, %}, {rbig, h}]
%[[2]]
example = example /. %
rbig[parts[example]["conical"]] * 2
radius[parts[example]["cylindrical"]] * 2
```

```
conicalTestTube[cylinder[14.81 - h, rbig], invertedFrustum[h, rbig, 0.152716, apexangle], cylinder[0, 0]]
```

$0.0248078\ (h - 6.4971\ rbig)^3 + (14.81 - h)\ \pi\ rbig^2 + 6.80375\ rbig^3 == 200$

$-0.153915\ h + rbig == 1.32$

Solve: Solve was unable to solve the system with inexact coefficients. The answer was obtained by solving a corresponding exact system and numericizing the result.

```
{{rbig → -1.51406, h → -18.4132}, {rbig → 2.23957, h → 5.97455}, {rbig → 4.6737, h → 21.7893}}
```

```
{rbig → 2.23957, h → 5.97455}
```

```
conicalTestTube[cylinder[8.83545, 2.23957], invertedFrustum[5.97455, 2.23957, 0.152716, apexangle], cylinder[0, 0]]
```

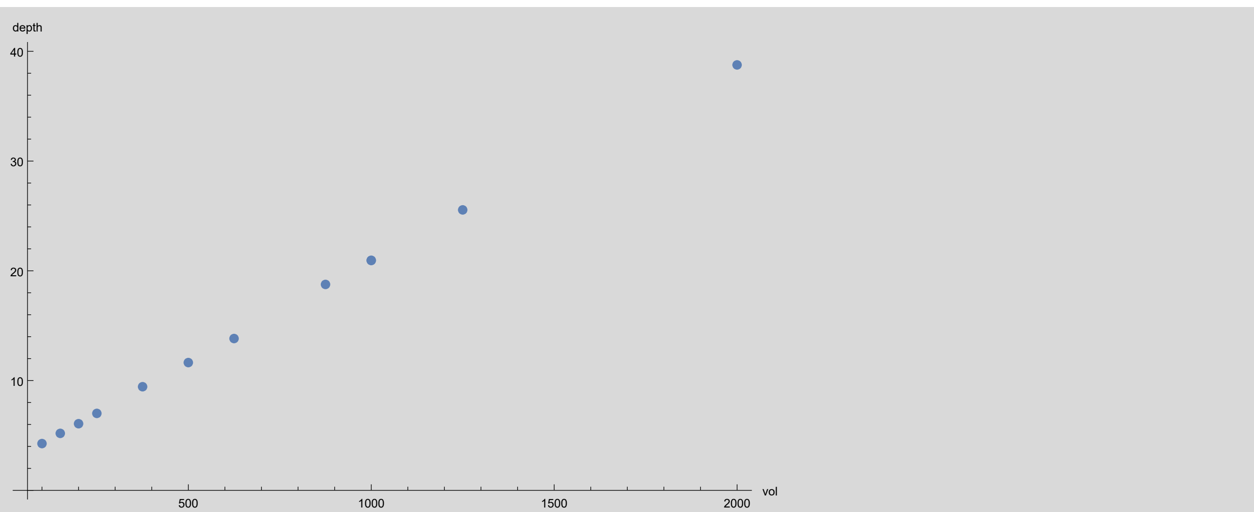
```
4.47914
```

```
4.47914
```

IDT tubes

```
idtData = ArrayReshape[{250, 7.01, 200, 6.07, 150, 5.19, 100, 4.26, 1000, 20.94,
   2000, 38.76, 1000, 20.96, 500, 11.64, 375, 9.44, 625, 13.83, 1250, 25.55, 875, 18.76}, {12, 2}]
ListPlot[idtData, ImageSize → Large, AxesLabel → {"vol", "depth"}, PlotRange → All]
```

```
{{250, 7.01}, {200, 6.07}, {150, 5.19}, {100, 4.26}, {1000, 20.94}, {2000, 38.76},
 {1000, 20.96}, {500, 11.64}, {375, 9.44}, {625, 13.83}, {1250, 25.55}, {875, 18.76}}
```



```
fitIdtData[data_] := Module[{depthFunc, cylinderData, vMin, hMin, offsetCylinderData, hCone, hCyl1, hCyl2,
   hCyl, rCyl, conePart, cylinderPart, errors, err, min, cylinderRules, tube, tubeRules, hOverall, idtRules},
  depthFunc[part_] := Module[{expr, v},
    expr = depthFromVolume[part, v];
    depthFunc[part] = Function[{vol}, expr /. {v → vol}]
   ];
  (* figure out the common radius of the cylinder & cone *)
  cylinderData = Select[data, True &];
  vMin = Min @ cylinderData[[All, 1]];
  hMin = Min @ cylinderData[[All, 2]];
  offsetCylinderData = {#[[1]] - vMin, #[[2]] - hMin} & /@ cylinderData;
  cylinderPart = cylinder[hCyl1, rCyl];
  errors = Function[{vol, depth},
       (depthFunc[cylinderPart][vol] - depth) ^ 2
      ] @@ # & /@ offsetCylinderData;
  err = Total[errors] // N;
  {min, cylinderRules} = NMinimize[{err, assumptions[cylinderPart] }, {hCyl1, rCyl}];
  test @ cylinderRules;

  (* figure out the height of the cone *)
  cylinderPart = cylinder[hCyl2, rCyl];
  conePart = invertedCone[hCone, rCyl];
  tube = conicalTestTube[cylinderPart, conePart, emptyCylinder[]] /. cylinderRules;
  test @ tube;
  errors = Function[{vol, depth},
       (depthFunc[tube][vol] - depth) ^ 2
      ] @@ # & /@ data;
  err = Total[errors] // N;
  {min, tubeRules} = NMinimize[{err }, {hCyl2, hCone}];
  test @ tubeRules;

  (* finally figure out the real height of the cylinder *)
  hOverall = 42; (* from opentrons labware *)
  tube = conicalTestTube[cylinder[hOverall - hCone, rCyl], conePart, emptyCylinder[]] /. cylinderRules /. tubeRules;
  tube
 ]
fittedIdt = fitIdtData[idtData]
test @ volume @ fittedIdt;
```

```
cylinderRules$41073 → {hCyl1$41073 → 6.4908, rCyl$41073 → 4.16389}
```

```
tube$41073 → conicalTestTube[cylinder[hCyl2$41073, 4.16389], invertedCone[hCone$41073, 4.16389], cylinder[0, 0]]
```

```
tubeRules$41073 → {hCyl2$41073 → 1.98558, hCone$41073 → 3.69629}
```

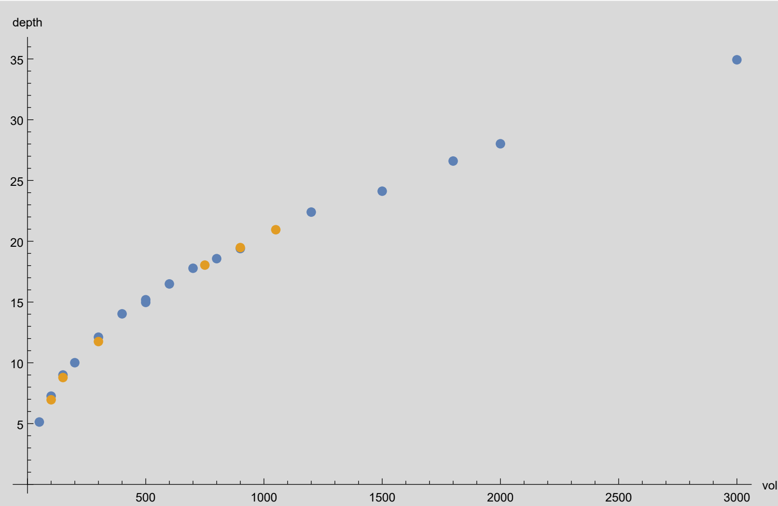
```
conicalTestTube[cylinder[38.3037, 4.16389], invertedCone[3.69629, 4.16389], cylinder[0, 0]]
```

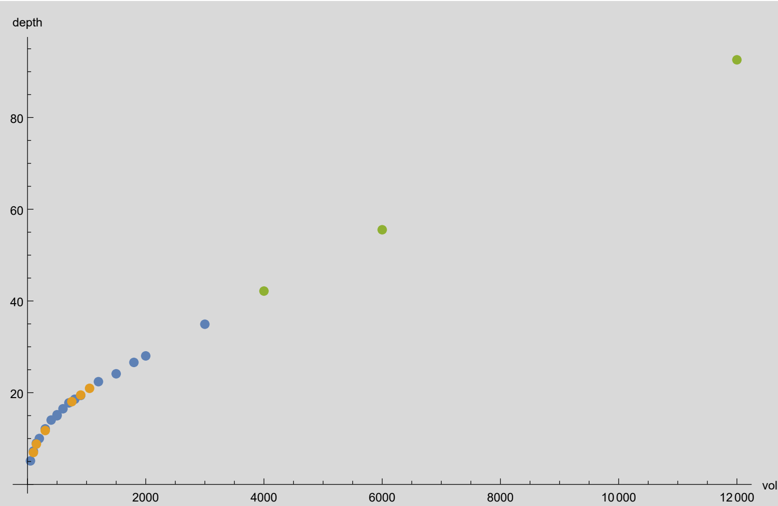
```
volume[fittedIdt] → 2153.47
```

Falcon

We have some empirical data for the 15mL Falcon tube.

```
Block[{hBase = 34.93},
  goodFalconData = {
    (*{1000, 19.78},*) {2000, 28.02}, {3000, hBase}, {500, 15.19}, (*{1000, 19.99},*) {50, 5.13}, {100, 7.26},
    {200, 10.01}, {150, 9.00}, {300, 12.11}, {600, 16.49}, {1200, 22.40}, {1800, 26.60},
    {400, 14.03}, {500, 14.97}, {700, 17.78}, {800, 18.57}, {900, 19.40}, {1500, 24.12}
   };
  okFalconData = {
    {100, 6.96}, {150, 8.79}, {300, 11.75}, (*{450, 14.32},*)
    (*{600, 15.89},*) {750, 18.04}, {900, 19.48}, {1050, 20.95}(*, {1200, 20.51}*)
   };
  upperFalconData = {
    {4000, hBase + 7.23}, {6000, hBase + 20.60} , {12000, hBase + 57.66}
   }];
ListPlot[{goodFalconData, okFalconData}, ImageSize → Large, AxesLabel → {"vol", "depth"}, PlotRange → All]
ListPlot[{goodFalconData, okFalconData, upperFalconData}, ImageSize → Large, AxesLabel → {"vol", "depth"}, PlotRange → All]
falconData = Union[goodFalconData ~Join~ okFalconData ~Join~ upperFalconData]
conicalFalconData = Select[falconData, #[[1]] ≤ 875 &]
```






```
{{50, 5.13}, {100, 6.96}, {100, 7.26}, {150, 8.79}, {150, 9.}, {200, 10.01}, {300, 11.75}, {300, 12.11}, {400, 14.03},
 {500, 14.97}, {500, 15.19}, {600, 16.49}, {700, 17.78}, {750, 18.04}, {800, 18.57}, {900, 19.4}, {900, 19.48}, {1050, 20.95},
 {1200, 22.4}, {1500, 24.12}, {1800, 26.6}, {2000, 28.02}, {3000, 34.93}, {4000, 42.16}, {6000, 55.53}, {12000, 92.59}}
```

```
{{50, 5.13}, {100, 6.96}, {100, 7.26}, {150, 8.79}, {150, 9.}, {200, 10.01}, {300, 11.75},
 {300, 12.11}, {400, 14.03}, {500, 14.97}, {500, 15.19}, {600, 16.49}, {700, 17.78}, {750, 18.04}, {800, 18.57}}
```

```
fitFalconData[data_] := Module[
  {threshold, conicalData, cylinderData, conePart, genericDepth, hCone, rmid, rbottom,
```

```
  errors, err, min, coneRules, angledCone, cylinderPart, hCyl, rtop, cylinderRules, angledCylinder,
  Δvol, Δh, vMin, hMin, offsetCylinderData, falcon, α, fassumpts, falconRules, first, second, hTot},

 (* first, fit the cone. this gives us the apex angle and rbottom *)
 conicalData = Select[data, #[[1]] ≤ 1000 &];
 conePart = invertedFrustum[hCone, rmid, rbottom];
 genericDepth[part_] := Module[{expr, v},
   expr = depthFromVolume[part, v];
   genericDepth[part] = Function[{vol}, expr /. {v → vol}]
  ];
 errors = Function[{vol, depth},
      (genericDepth[conePart][vol] - depth) ^ 2
     ] @@ # & /@ conicalData;
 err = Total[errors] // N;
 {min, coneRules} = NMinimize[{err, assumptions[conePart] && hCone > 15}, {hCone, rmid, rbottom}];
 angledCone = toApexAngled[conePart /. coneRules];

 (* now for the cylinder. this gives us the apex angle *)
 cylinderData = Select[data, #[[1]] ≥ 1200 &]; (* hard to tell for in between data, so we're conservative *)
 vMin = Min @ cylinderData[[All, 1]];
 hMin = Min @ cylinderData[[All, 2]];
 offsetCylinderData = {#[[1]] - vMin, #[[2]] - hMin} & /@ cylinderData;
 cylinderPart = invertedFrustum[hCyl, rtop, rmid] /. coneRules;
 errors = Function[{vol, depth},
      (genericDepth[cylinderPart][vol] - depth) ^ 2
     ] @@ # & /@ offsetCylinderData;
 err = Total[errors] // N;
 {min, cylinderRules} = NMinimize[{err, assumptions[cylinderPart] }, {hCyl, rtop}];
 angledCylinder = toApexAngled[cylinderPart /. cylinderRules];

 falcon = conicalTestTube[
   (invertedFrustum[hCyl, hCyl Tan[α] + rmid, α, apexangle] /. {α → apexangle[angledCylinder]}),
   (invertedFrustum[hCone, hCone Tan[α] + rbottom, α, apexangle] /.
     {α → apexangle[angledCone], rbottom → (rbottom /. coneRules)}),
   emptyCylinder[]
  ];
 fassumpts = hCone > 18 && hCone < 24.5 && rmid > 6 && hCyl > 75;
 hTot = 119.46 - 1.39;
 errors = Function[{vol, depth},
      (FullSimplify[genericDepth[falcon][vol] - depth, fassumpts]) ^ 2
     ] @@ # & /@ data;
 err = Total[errors] // N;

 (* put together to get rmid, hCyl, and hCone*)
 first[] := Module[{},
   {min, falconRules} = NMinimize[{err, fassumpts }, {hCyl, hCone, rmid}];
   test @ (falcon /. falconRules);
   Function[f, conicalTestTube[
      toCartesian[parts[f]["cylindrical"]],
      toCartesian[parts[f]["conical"]],
      emptyCylinder[]
     ]][falcon /. falconRules]
  ];
 second[] := Module[{rule = hCyl → hTot - hCone},
   {min, falconRules} = NMinimize[{err /. rule, fassumpts /. rule }, {hCone, rmid}];
   test @ (falcon /. falconRules);
   Function[f, conicalTestTube[
      toCartesian[parts[f]["cylindrical"]],
      toCartesian[parts[f]["conical"]],
      emptyCylinder[]
     ]][falcon /. rule /. falconRules]
  ];
 {first[], second[]}
]
```

```
{fittedFalcon1, fittedFalcon2} = fitFalconData[falconData];
fittedFalcon1
fittedFalcon2
fittedFalcon = fittedFalcon2;
test @ volume[fittedFalcon];
test @ depthFromVolume[fittedFalcon, volume[fittedFalcon]];
```

```
(falcon$41707 /. falconRules$41707) → conicalTestTube[invertedFrustum[76.8592, 7.27546, 0.00805924, apexangle],
  invertedFrustum[22.0945, 6.65602, 0.244311, apexangle], cylinder[0, 0]]
```

```
(falcon$41707 /. falconRules$41707) →
 conicalTestTube[invertedFrustum[hCyl$41707, 6.65602 + 0.00805941 hCyl$41707, 0.00805924, apexangle],
  invertedFrustum[22.0945, 6.65602, 0.244311, apexangle], cylinder[0, 0]]
```

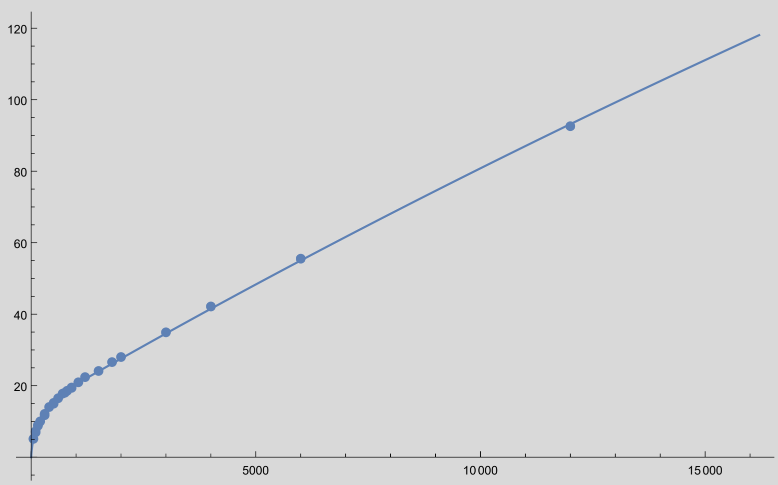
```
conicalTestTube[invertedFrustum[76.8592, 7.27546, 6.65602], invertedFrustum[22.0945, 6.65602, 1.14806], cylinder[0, 0]]
```

```
conicalTestTube[invertedFrustum[95.9755, 7.42952, 6.65602], invertedFrustum[22.0945, 6.65602, 1.14806], cylinder[0, 0]]
```

```
volume[fittedFalcon] → 16 202.8
```

```
depthFromVolume[fittedFalcon, volume[fittedFalcon]] → 118.07
```

```
Block[{expr},
 expr = depthFromVolume[fittedFalcon, vol];
 Show[
  Plot[expr, {vol, 0, volume[fittedFalcon]}, ImageSize → Large],
  ListPlot[falconData]]]
```



Known Tubes

Definitions

With that, we define the tubes

```
(tubes = {
    (* we ignore the slight widening at the throat. and the bottom cap isn't a complete hemi-sphere,
    though we treat it as such *)
    eppendorf5$0mL → Block[{side = 56.7 - 55.4, hTop = 34.12 + 2.2},
      toCanonical @ conicalTestTube[{14.8, 13.3, 3.3}, {hTop, 55.4 - hTop}]],

    eppendorf1$5ml → Block[{wall = (*measured@1000*) 10.34 - 8.81, hTop = 20},
      toCanonical @ conicalTestTube[{9.0 (*measured*), 8.7, 3.6}, {hTop, 37.8 - hTop}]],
    fittedEppendorf1$5ml → fittedEppendorf,

    fittedFalcon15ml → fittedFalcon,
    falcon15ml → Module[
      (* mixure of measurements and values from spec drawing *)
      (* FWIW, Opentrons uses idTop=14.9, depth=117.5. The latter is pretty good,
      given 'a' and 'wall' defined here, so our depth calc's should be good *)
      {id14, od14, wall14, wallMeasured, wall, a, b, a14, b14, c, cMeasured, d,
       bottomOd, wallCap, htopMeasured, hBottomAndCap},
      id14 = 15.0;
      od14 = 16.3;
      wall14 = od14 - id14;
      wallMeasured = 1.27;
      wall = wallMeasured;
      wallCap = 1.75;
      a = 118.8;
      b = 17.37;
      a14 = 106.3;
      b14 = 16.6;
      c = 15.75;
      cMeasured = 15.1;
      d = 22.48;
      bottomOd = 3.18;
      htopMeasured = 84.07;
      hBottomAndCap = d - wallCap;
      (* note: as defined here, we only have 14mL capacity, not 15mL. Will affect volume calc but not depth calc. *)
      toCanonical @ conicalTestTube[{b14 - (*2 - logically needed, but better fit w/o (?!)*) wall,
         cMeasured - 2 wall, bottomOd - 2 wall}, {htopMeasured, hBottomAndCap}]
     ],
    generic → toCanonical @ conicalTestTube[{idTop, idHip, idBottom}, {hTop, hBottom}],

    (* this hacks in the slightly shallower taper at the top, which isn't sized on the spec drawing *)
    bioradPlateWell → Module[{hCyl = 0.15, rbig = 5.46 / 2, rsmall = 2.64 / 2, cyl, con, cap},
      cyl = cylinder[hCyl, rbig];
      con = invertedFrustum[14.81 - hCyl, rbig, rsmall];
      cap = emptyCylinder[];
      conicalTestTube[cyl, con, cap]],

    (* see above *)
    bioradPlateWell2 → conicalTestTube[cylinder[8.835453539401207`, 2.239570651942052`],
      invertedFrustum[5.974546460598792`, 2.239570651942052`, 0.15271630954950383`, apexangle], cylinder[0, 0]],

    idtTube → conicalTestTube[
      cylinder[40.73, 8.31 / 2],
      invertedCone[3.2, 8.31 / 2],
      emptyCylinder[]
     ],
    fittedIdtTube → fittedIdt
   } // Association) // Normal // ColumnForm
```

```
eppendorf5$0mL → conicalTestTube[invertedFrustum[36.32, 7.4, 6.65], invertedFrustum[15.78, 6.65, 1.65], invertedSphericalCap[1.65, 1.(
eppendorf1$5ml → conicalTestTube[invertedFrustum[20, 4.5, 4.35], invertedFrustum[14.2, 4.35, 1.8], invertedSphericalCap[1.8, 1.8]]
fittedEppendorf1$5ml → conicalTestTube[invertedFrustum[18.9894, 4.70751, 4.35636], invertedFrustum[16.8419, 4.35636, 2.1099], unknown
fittedFalcon15ml → conicalTestTube[invertedFrustum[95.9755, 7.42952, 6.65602], invertedFrustum[22.0945, 6.65602, 1.14806], cylinder[0
falcon15ml → conicalTestTube[invertedFrustum[84.07, 7.665, 6.28], invertedFrustum[20.09, 6.28, 0.32], invertedSphericalCap[0.32, 0.32
generic → conicalTestTube[invertedFrustum[hTop, idTop/2, idHip/2], invertedFrustum[hBottom - idBottom, idHip/2, idBottom/2], invertedSphericalCap[
bioradPlateWell → conicalTestTube[cylinder[0.15, 2.73], invertedFrustum[14.66, 2.73, 1.32], cylinder[0, 0]]
bioradPlateWell2 → conicalTestTube[cylinder[8.83545, 2.23957], invertedFrustum[5.97455, 2.23957, 0.152716, apexangle], cylinder[0, 0]
idtTube → conicalTestTube[cylinder[40.73, 4.155], invertedCone[3.2, 4.155], cylinder[0, 0]]
fittedIdtTube → conicalTestTube[cylinder[38.3037, 4.16389], invertedCone[3.69629, 4.16389], cylinder[0, 0]]
```

Calibrating against known tubes

```
test @ depthFromVolume[tubes[eppendorf1$5ml], 500];
test @ depthFromVolume[tubes[eppendorf1$5ml], 1500];
test @ (depthFromVolume[tubes[eppendorf1$5ml], 1500] - depthFromVolume[tubes[eppendorf1$5ml], 1000]);
```

```
depthFromVolume[tubes[eppendorf1$5ml], 500] → 16.7021
```

```
depthFromVolume[tubes[eppendorf1$5ml], 1500] → 33.0204
```

```
depthFromVolume[tubes[eppendorf1$5ml], 1500] - depthFromVolume[tubes[eppendorf1$5ml], 1000] → 8.0461
```

```
test @ depthFromVolume[tubes[fittedEppendorf1$5ml], 500];
test @ depthFromVolume[tubes[fittedEppendorf1$5ml], 1500];
test @ (depthFromVolume[tubes[fittedEppendorf1$5ml], 1500] - depthFromVolume[tubes[eppendorf1$5ml], 1000]);
```

```
depthFromVolume[tubes[fittedEppendorf1$5ml], 500] → 17.4848
```

```
depthFromVolume[tubes[fittedEppendorf1$5ml], 1500] → 33.3897
```

```
depthFromVolume[tubes[fittedEppendorf1$5ml], 1500] - depthFromVolume[tubes[eppendorf1$5ml], 1000] → 8.41539
```

```
test @ depthFromVolume[tubes[eppendorf5$0mL], 5000];
```

```
depthFromVolume[tubes[eppendorf5$0mL], 5000] → 44.1795
```

```
test @ tubes[falcon15ml];
test @ depthFromVolume[tubes[falcon15ml], 3000];
test @ depthFromVolume[tubes[falcon15ml], 14000];
test @ (depthFromVolume[tubes[falcon15ml], 14000] - depthFromVolume[tubes[falcon15ml], 2000](* measured at 76.5*));
```

```
tubes[falcon15ml] →
 conicalTestTube[invertedFrustum[84.07, 7.665, 6.28], invertedFrustum[20.09, 6.28, 0.32], invertedSphericalCap[0.32, 0.32]]
```

```
depthFromVolume[tubes[falcon15ml], 3000] → 36.8483
```

```
depthFromVolume[tubes[falcon15ml], 14000] → 105.795
```

```
depthFromVolume[tubes[falcon15ml], 14000] - depthFromVolume[tubes[falcon15ml], 2000] → 76.5075
```

```
test @ tubes[fittedFalcon15ml];
test @ depthFromVolume[tubes[fittedFalcon15ml], 3000];
test @ depthFromVolume[tubes[fittedFalcon15ml], 14000];
test @
  (depthFromVolume[tubes[fittedFalcon15ml], 14000] - depthFromVolume[tubes[fittedFalcon15ml], 2000](* measured at 76.5*));
```

```
tubes[fittedFalcon15ml] →
 conicalTestTube[invertedFrustum[95.9755, 7.42952, 6.65602], invertedFrustum[22.0945, 6.65602, 1.14806], cylinder[0, 0]]
```

```
depthFromVolume[tubes[fittedFalcon15ml], 3000] → 34.6045
```

```
depthFromVolume[tubes[fittedFalcon15ml], 14000] → 105.188
```

```
depthFromVolume[tubes[fittedFalcon15ml], 14000] - depthFromVolume[tubes[fittedFalcon15ml], 2000] → 77.6146
```

```
test @ tubes[bioradPlateWell];
test @ depthFromVolume[tubes[bioradPlateWell], 84];
test @ depthFromVolume[tubes[bioradPlateWell], 84 - 50];
test @ toDeg @ apexangle @ parts[tubes[bioradPlateWell]]["conical"];
```

```
tubes[bioradPlateWell] → conicalTestTube[cylinder[0.15, 2.73], invertedFrustum[14.66, 2.73, 1.32], cylinder[0, 0]]
```

```
depthFromVolume[tubes[bioradPlateWell], 84] → 8.68692
```

```
depthFromVolume[tubes[bioradPlateWell], 84 - 50] → 4.54217
```

```
toDeg[apexangle[parts[tubes[bioradPlateWell]][conical]]] → 5.49381
```

```
test @ tubes[bioradPlateWell2];
test @ depthFromVolume[tubes[bioradPlateWell2], 84];
test @ depthFromVolume[tubes[bioradPlateWell2], 84 - 50];
test @ toDeg @ apexangle @ parts[tubes[bioradPlateWell2]]["conical"];
```

```
tubes[bioradPlateWell2] →
 conicalTestTube[cylinder[8.83545, 2.23957], invertedFrustum[5.97455, 2.23957, 0.152716, apexangle], cylinder[0, 0]]
```

```
depthFromVolume[tubes[bioradPlateWell2], 84] → 7.44829
```

```
depthFromVolume[tubes[bioradPlateWell2], 84 - 50] → 4.0258
```

```
toDeg[apexangle[parts[tubes[bioradPlateWell2]][conical]]] → 8.75
```

```
test @ depthFromVolume[tubes[idtTube], 250];
test @ (depthFromVolume[tubes[idtTube], 1250] - depthFromVolume[tubes[idtTube], 250]);
```

```
depthFromVolume[tubes[idtTube], 250] → 6.74277
```

```
depthFromVolume[tubes[idtTube], 1250] - depthFromVolume[tubes[idtTube], 250] → 18.4378
```

For volume as parameter

```
printAndPlot[name_] := Module[{expr},
  CellPrint[TextCell[name, "Text"]];
  If[ToString[name] == "generic",
   test @ depthFromVolume[tubes[name], vol];
   ,
   test @ N @ depthFromVolume[tubes[name], vol];
   test @ N @ volume[tubes[name]];
   test @ N @ depthFromVolume[tubes[name], volume[tubes[name]]];
   expr = N @ depthFromVolume[tubes[name], vol];
   printCell @
    Plot[expr, {vol, 0, volume[tubes[name]]}, AxesLabel → {"volume", "depth"}, PlotLabel → name, AxesOrigin → {0, 0}]
  ]]
printAndPlot /@ Keys[tubes];
```

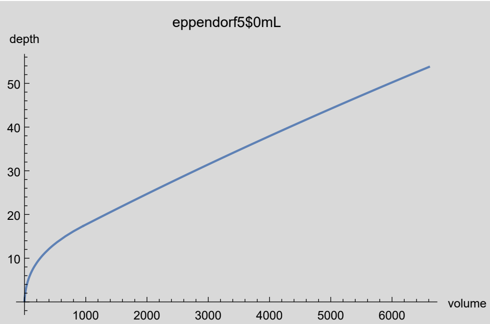
eppendorf5$0mL

N[depthFromVolume[tubes[eppendorf5$0mL], vol]] →

$$\begin{cases} 1.65 - \frac{2.51187-4.35069\,i}{\left(28.2249-3.\,vol+1.73205\sqrt{-56.4497\,vol+3.\,vol^2}\right)^{1/3}} - (0.270963+0.469322\,i)\left(28.2249-3.\,vol+1.73205\sqrt{-56.4497\,vol+3.\,vol^2}\right)^{1/3} & vol \leq 9.40828 \\ -3.5574+1.25825\,(25.9645+4.77465\,vol)^{1/3} & vol \leq 957.074 \\ -304.607+14.623\,(9988.78+0.716197\,vol)^{1/3} & \text{True} \end{cases}$$

N[volume[tubes[eppendorf5$0mL]]] → 6602.87

N[depthFromVolume[tubes[eppendorf5$0mL], volume[tubes[eppendorf5$0mL]]]] → 53.75



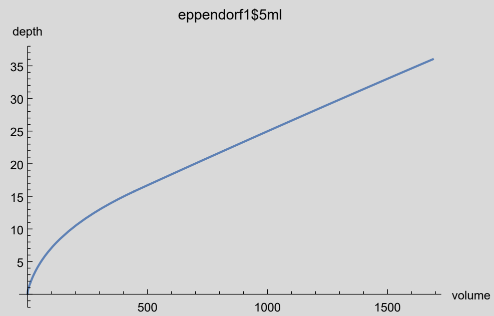
eppendorf1$5ml

N[depthFromVolume[tubes[eppendorf1$5ml], vol]] →

$$\begin{cases} 1.8 - \frac{2.98934-5.17768\,i}{\left(36.6435-3.\,vol+1.73205\sqrt{-73.2871\,vol+3.\,vol^2}\right)^{1/3}} - (0.270963+0.469322\,i)\left(36.6435-3.\,vol+1.73205\sqrt{-73.2871\,vol+3.\,vol^2}\right)^{1/3} & vol \leq 12.2145 \\ -8.22353+2.2996\,(53.0712+2.43507\,vol)^{1/3} & vol \leq 445.995 \\ -564.+49.1204\,(1580.62+0.143239\,vol)^{1/3} & \text{True} \end{cases}$$

N[volume[tubes[eppendorf1$5ml]]] → 1688.61

N[depthFromVolume[tubes[eppendorf1$5ml], volume[tubes[eppendorf1$5ml]]]] → 36.



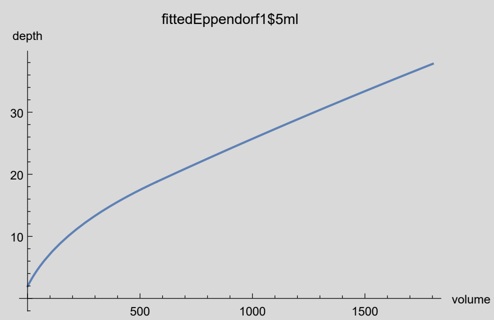

```
fittedEppendorf1$5ml
```

```
N[depthFromVolume[tubes[fittedEppendorf1$5ml], vol]] →
```

$$\begin{cases} \text{If}[\text{vol} \le 0., 0., \text{Indeterminate}] & \text{vol} \le 0.550217 \\ -13.8495 + 2.9248\ (157.009 + 2.14521\ \text{vol})^{1/3} & \text{vol} \le 575.33 \\ -216.767 + 20.2694\ (1376.83 + 0.33533\ \text{vol})^{1/3} & \text{True} \end{cases}$$

```
N[volume[tubes[fittedEppendorf1$5ml]]] → 1801.76
```

```
N[depthFromVolume[tubes[fittedEppendorf1$5ml], volume[tubes[fittedEppendorf1$5ml]]]] → 37.8
```



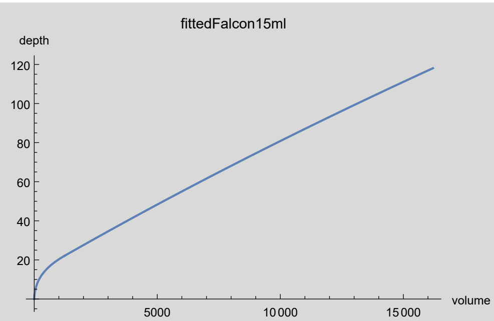

```
fittedFalcon15ml
```

```
N[depthFromVolume[tubes[fittedFalcon15ml], vol]] →
```

$$\begin{cases} 0. & \text{vol} \le 0. \\ -4.60531 + 1.42955\ (33.4335 + 5.25971\ \text{vol})^{1/3} & \text{vol} \le 1232.34 \\ -803.774 + 27.1004\ (27\,390.9 + 0.738644\ \text{vol})^{1/3} & \text{True} \end{cases}$$

```
N[volume[tubes[fittedFalcon15ml]]] → 16 202.8
```

```
N[depthFromVolume[tubes[fittedFalcon15ml], volume[tubes[fittedFalcon15ml]]]] → 118.07
```



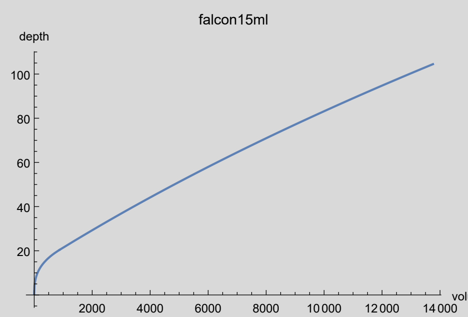

```
falcon15ml
```

N[depthFromVolume[tubes[falcon15ml], vol]] →

$$\begin{cases} 0.32 - \dfrac{0.0944778 - 0.16364\, i}{\left(0.205887 - 3.\, vol + 1.73205 \sqrt{-0.411775\, vol + 3.\, vol^2}\right)^{1/3}} - (0.270963 + 0.469322\, i) \left(0.205887 - 3.\, vol + 1.73205 \sqrt{-0.411775\, vol + 3.\, vol^2}\right)^{1/3} & vol \leq 0.0686291 \\ -0.758658 + 1.23996\, (0.267715 + 5.69138\, vol)^{1/3} & vol \leq 874.146 \\ -360.788 + 13.8562\, (19665.7 + 1.32258\, vol)^{1/3} & \text{True} \end{cases}$$

N[volume[tubes[falcon15ml]]] → 13756.5

N[depthFromVolume[tubes[falcon15ml], volume[tubes[falcon15ml]]]] → 104.48



generic

depthFromVolume[tubes[generic], vol] →

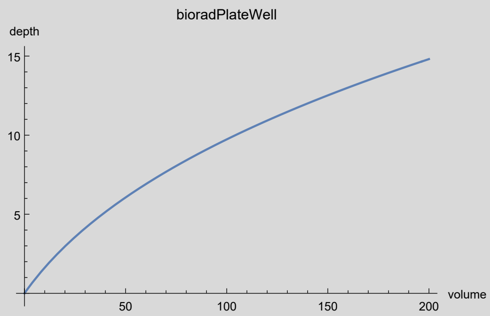
$$\begin{cases} \dfrac{idBottom}{2} - \dfrac{\left(1 - i\sqrt{3}\right) idBottom^2\, \pi^{1/3}}{4\, 2^{2/3} \left(\frac{idBottom^3 \pi}{4} - 3\, vol + \sqrt{3} \sqrt{-\frac{1}{2}\, idBottom^3\, \pi\, vol + 3\, vol^2}\right)^{1/3}} - \dfrac{\left(1 + i\sqrt{3}\right) \left(\frac{idBottom^3 \pi}{4} - 3\, vol + \sqrt{3} \sqrt{-\frac{1}{2}\, idBottom^3\, \pi\, vol + 3\, vol^2}\right)^{1/3}}{2\, (2\pi)^{1/3}} & vol \leq \dfrac{idBottom^3\, \pi}{12} \\ \dfrac{idBottom}{2} - \dfrac{1}{idBottom - idHip} \left(-hBottom\, idBottom + idBottom^2 + (hBottom - idBottom)^{2/3} \left(idBottom^3\, (hBottom - idHip) + \frac{12\, (-idBottom + idHip)\, vol}{\pi}\right)^{1/3}\right) & vol \leq \frac{1}{12}\, (hBottom - idBottom) \left(idBottom^2 + idBottom\, idHip + idHip^2\right) \pi \\ hBottom - \dfrac{idBottom}{2} + \dfrac{1}{idHip - idTop} \left(hTop\, idHip - hTop^{2/3} \left(hBottom \left(idBottom^2 + idBottom\, idHip + idHip^2\right) (idHip - idTop) + idHip\, (idHip\, (hTop\, idHip - idBottom\, (idBottom + idHip)) + idBottom\, (idBottom + idHip)\, idTop) + \frac{12\, (-idHip + idTop)\, vol}{\pi}\right)^{1/3}\right) & \text{Tr} \end{cases}$$

bioradPlateWell

N[depthFromVolume[tubes[bioradPlateWell], vol]] → $\begin{cases} 0. & vol \leq 0. \\ -13.7243 + 4.24819\, (33.7175 + 1.34645\, vol)^{1/3} & vol \leq 196.488 \\ 14.66 - 0.0427095\, (196.488 - 1.\, vol) & \text{True} \end{cases}$

N[volume[tubes[bioradPlateWell]]] → 200.

N[depthFromVolume[tubes[bioradPlateWell], volume[tubes[bioradPlateWell]]]] → 14.81

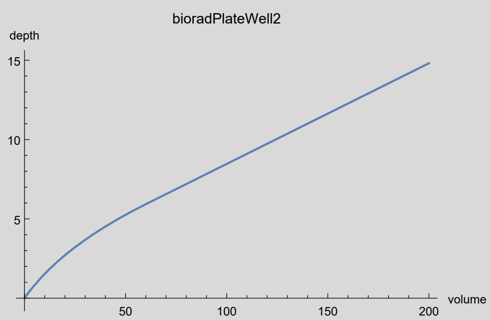


bioradPlateWell2

$$\texttt{N[depthFromVolume[tubes[bioradPlateWell2], vol]]} \to \begin{cases} 0. & \text{vol} \le 0. \\ -8.57618 + 6.4971\,(2.29997 + 0.146978\,\text{vol})^{1/3} & \text{vol} \le 60.7779 \\ 5.97455 - 0.063463\,(60.7779 - 1.\,\text{vol}) & \text{True} \end{cases}$$

`N[volume[tubes[bioradPlateWell2]]]` → 200.

`N[depthFromVolume[tubes[bioradPlateWell2], volume[tubes[bioradPlateWell2]]]]` → 14.81

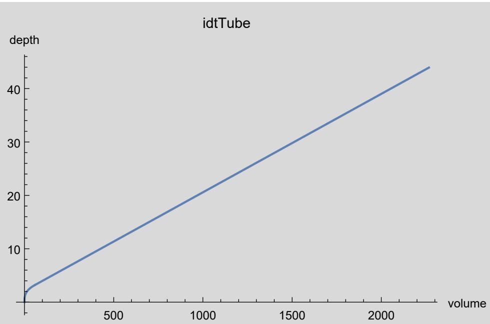


idtTube

$$\texttt{N[depthFromVolume[tubes[idtTube], vol]]} \to \begin{cases} 0. & \text{vol} \le 0. \\ 0.827389\,\text{vol}^{1/3} & \text{vol} \le 57.8523 \\ 3.2 - 0.0184378\,(57.8523 - 1.\,\text{vol}) & \text{True} \end{cases}$$

`N[volume[tubes[idtTube]]]` → 2266.91

`N[depthFromVolume[tubes[idtTube], volume[tubes[idtTube]]]]` → 43.93

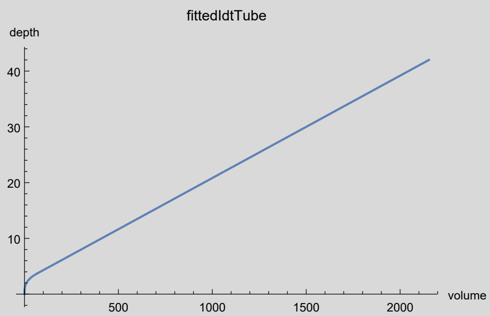


fittedIdtTube

$$\texttt{N[depthFromVolume[tubes[fittedIdtTube], vol]]} \to \begin{cases} 0. & \text{vol} \le 0. \\ 0.909568\,\text{vol}^{1/3} & \text{vol} \le 67.1109 \\ 3.69629 - 0.0183591\,(67.1109 - 1.\,\text{vol}) & \text{True} \end{cases}$$

```
N[volume[tubes[fittedIdtTube]]] → 2153.47
```

```
N[depthFromVolume[tubes[fittedIdtTube], volume[tubes[fittedIdtTube]]]] → 42.
```



Comparing 1.5 mL Eppendorf Tube Models

The fitted Eppendorf model clearly is better.

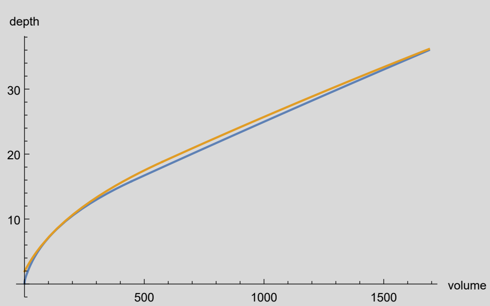
```
example1 = tubes[eppendorf1$5ml];
example2 = tubes[fittedEppendorf1$5ml];
test @ example1;
test @ example2;
expr1 = depthFromVolume[example1, v]
expr2 = depthFromVolume[example2, v]
Plot[{expr1, expr2}, {v, 0, volume[example1]}, AxesLabel → {"volume", "depth"}]
Plot[expr1 - expr2, {v, 0, volume[example1]}, AxesLabel → {"volume", "Δdepth"}]
Show[ListPlot[{eppendorfData}, AxesLabel → {"vol", "depth"}, PlotRange → All, AxesOrigin → {0, 0}, ImageSize → Large],
 Plot[{depthFromVolume[example1, v], depthFromVolume[example2, v]}, {v, 0, volume[example1]}]]
```

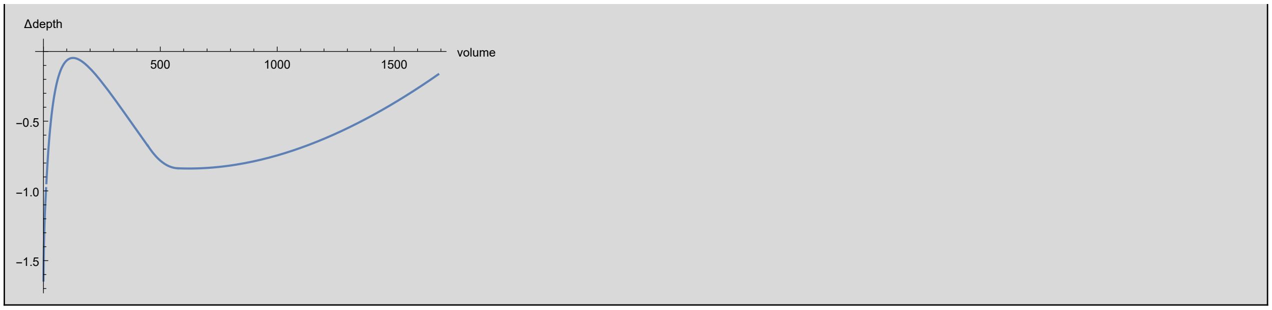
```
example1 → conicalTestTube[invertedFrustum[20, 4.5, 4.35], invertedFrustum[14.2, 4.35, 1.8], invertedSphericalCap[1.8, 1.8]]
```

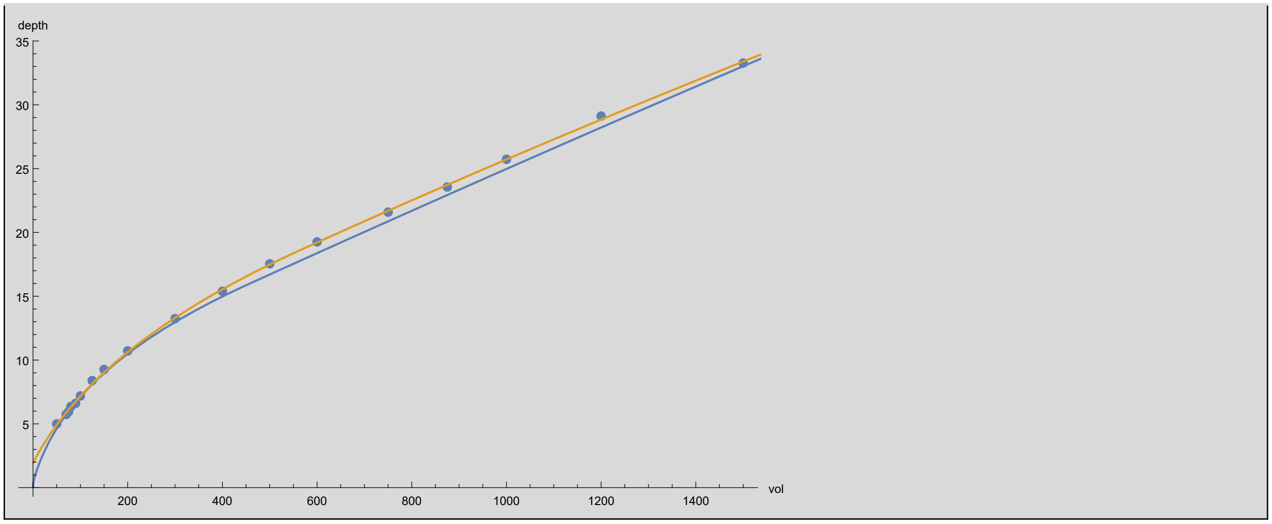
```
example2 → conicalTestTube[invertedFrustum[18.9894, 4.70751, 4.35636],
  invertedFrustum[16.8419, 4.35636, 2.1099], unknownShape[1.96866, 0.550217]]
```

$$\begin{cases} 1.8 - \dfrac{2.98934 - 5.17768\, i}{\left(36.6435 - 3\, v + \sqrt{3}\, \sqrt{-73.2871\, v + 3\, v^2}\right)^{1/3}} - \dfrac{\left(1 + i\sqrt{3}\right)\left(36.6435 - 3\, v + \sqrt{3}\, \sqrt{-73.2871\, v + 3\, v^2}\right)^{1/3}}{2\, (2\, \pi)^{1/3}} & v \le 12.2145 \\ -8.22353 + 2.2996\, (53.0712 + 2.43507\, v)^{1/3} & v \le 445.995 \\ -564. + 49.1204\, (1580.62 + 0.143239\, v)^{1/3} & \text{True} \end{cases}$$

$$\begin{cases} \text{If}[v \le 0, 0, \text{Indeterminate}] & v \le 0.550217 \\ -13.8495 + 2.9248\, (157.009 + 2.14521\, v)^{1/3} & v \le 575.33 \\ -216.767 + 20.2694\, (1376.83 + 0.33533\, v)^{1/3} & \text{True} \end{cases}$$







Comparing IDT Tube Models

The fitted IDT tube model is marginally better, but still better.

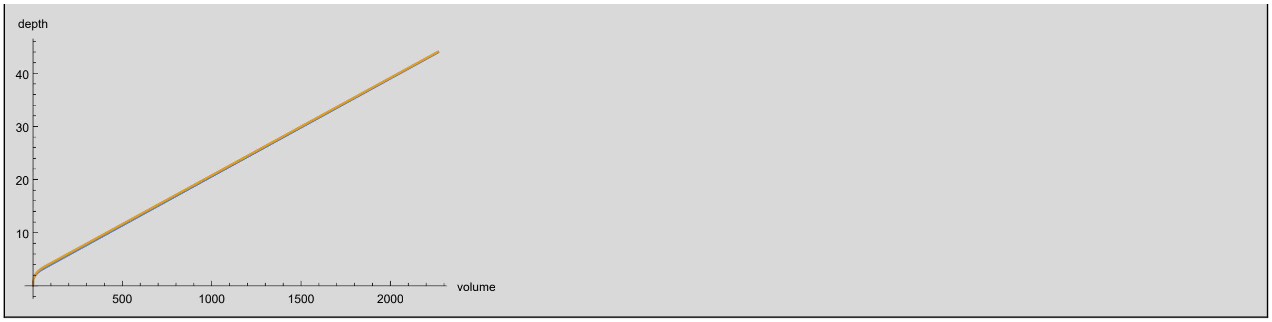
```
example1 = tubes[idtTube];
example2 = tubes[fittedIdtTube];
test @ example1;
test @ example2;
expr1 = depthFromVolume[example1, v]
expr2 = depthFromVolume[example2, v]
Plot[{expr1, expr2}, {v, 0, volume[example1]}, AxesLabel → {"volume", "depth"}]
Plot[expr1 - expr2, {v, 0, volume[example1]}, AxesLabel → {"volume", "Δdepth"}]
Show[ListPlot[{idtData}, AxesLabel → {"vol", "depth"}, PlotRange → All, AxesOrigin → {0, 0}, ImageSize → Large],
 Plot[{depthFromVolume[example1, v], depthFromVolume[example2, v]}, {v, 0, volume[example1]}]]
```

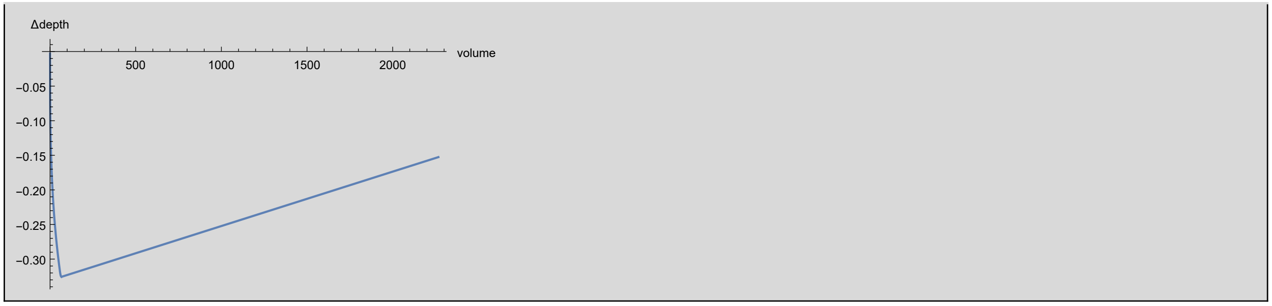
```
example1 → conicalTestTube[cylinder[40.73, 4.155], invertedCone[3.2, 4.155], cylinder[0, 0]]
```

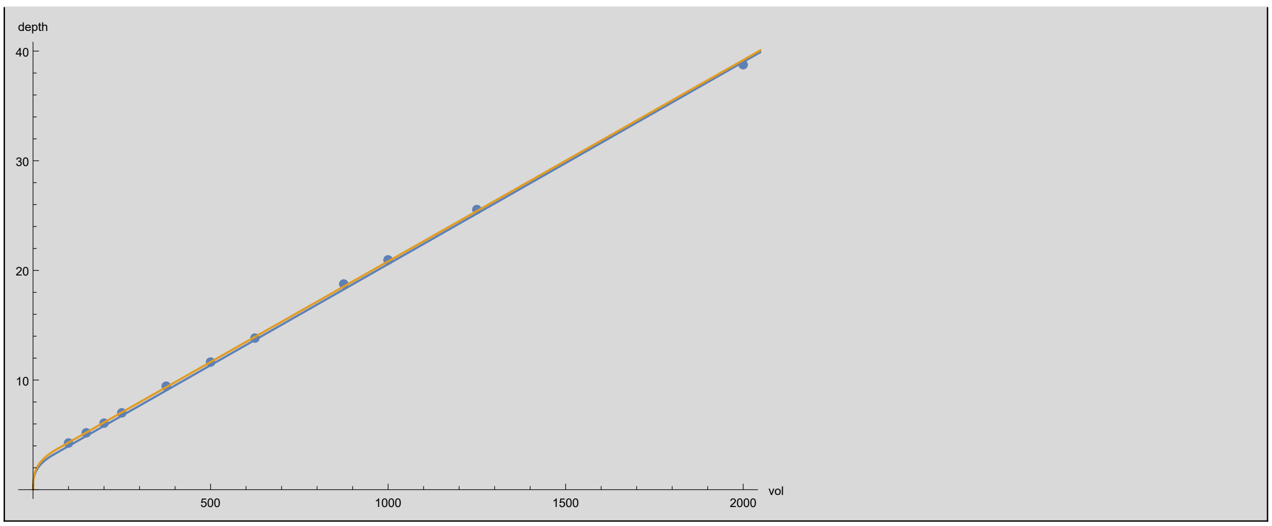
```
example2 → conicalTestTube[cylinder[38.3037, 4.16389], invertedCone[3.69629, 4.16389], cylinder[0, 0]]
```

$$\begin{cases} 0 & v \le 0 \\ 0.827389\, v^{1/3} & v \le 57.8523 \\ 3.2 - 0.0184378\,(57.8523 - v) & \text{True} \end{cases}$$

$$\begin{cases} 0 & v \le 0 \\ 0.909568\, v^{1/3} & v \le 67.1109 \\ 3.69629 - 0.0183591\,(67.1109 - v) & \text{True} \end{cases}$$







Comparing Bio-rad Plate models

Which should we use? At the moment it’s unclear.

```
example1 = tubes[bioradPlateWell];
example2 = tubes[bioradPlateWell2]; (* currently in use *)
examplem1 = modelBioRad1[]; (*same as example 1*)
examplem2 = modelBioRad2[];
examplem3 = modelBioRad3[];
test @ example1;
test @ example2;
test @ examplem1;
test @ examplem2;
test @ examplem3;
expr1 = depthFromVolume[example1, v]
expr2 = depthFromVolume[example2, v]
exprm1 = depthFromVolume[examplem1, v]
exprm2 = depthFromVolume[examplem2, v]
exprm3 = depthFromVolume[examplem3, v]
Plot[{expr1, expr2, exprm1, exprm2, exprm3}, {v, 0, volume[examplem3]},
 AxesLabel → {"volume", "depth"}, PlotLegends → Automatic, GridLines → Automatic]
Plot[{expr2 - expr1, expr2 - expr2, expr2 - exprm1, expr2 - exprm2, expr2 - exprm3}, {v, 0, volume[examplem3]},
 AxesLabel → {"volume", "Δdepth"}, PlotLegends → Automatic, PlotRange → All, GridLines → Automatic]
```

```
example1 → conicalTestTube[cylinder[0.15, 2.73], invertedFrustum[14.66, 2.73, 1.32], cylinder[0, 0]]
```

```
example2 → conicalTestTube[cylinder[8.83545, 2.23957], invertedFrustum[5.97455, 2.23957, 0.152716, apexangle], cylinder[0, 0]]
```

```
examplem1 → conicalTestTube[cylinder[0.150026, 2.73], invertedFrustum[14.66, 2.73, 1.32], cylinder[0, 0]]
```

```
examplem2 → conicalTestTube[cylinder[2.83192, 2.73], invertedFrustum[11.9781, 2.73, 0.886397], cylinder[0, 0]]
```

```
examplem3 → conicalTestTube[cylinder[5.64908, 2.73], invertedFrustum[9.16092, 2.73, 1.32], cylinder[0, 0]]
```

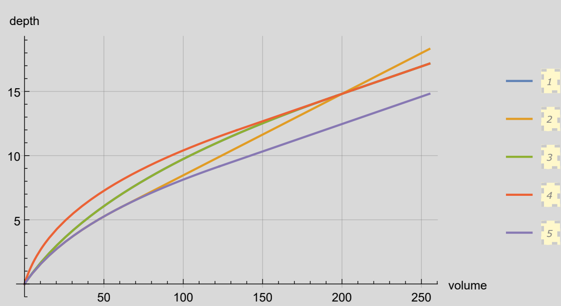
$$\begin{cases} 0 & v \le 0 \\ -13.7243 + 4.24819\,(33.7175 + 1.34645\,v)^{1/3} & v \le 196.488 \\ 14.66 - 0.0427095\,(196.488 - v) & \text{True} \end{cases}$$

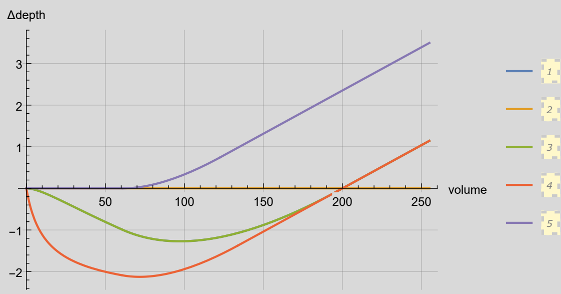
$$\begin{cases} 0 & v \le 0 \\ -8.57618 + 6.4971\,(2.29997 + 0.146978\,v)^{1/3} & v \le 60.7779 \\ 5.97455 - 0.063463\,(60.7779 - v) & \text{True} \end{cases}$$

$$\begin{cases} 0 & v \le 0 \\ -13.7242 + 4.24818\,(33.7175 + 1.34645\,v)^{1/3} & v \le 196.487 \\ 14.66 - 0.0427095\,(196.487 - v) & \text{True} \end{cases}$$

$$\begin{cases} 0 & v \le 0 \\ -5.75901 + 2.8396\,(8.34203 + 1.76051\,v)^{1/3} & v \le 133.694 \\ 11.9781 - 0.0427095\,(133.694 - v) & \text{True} \end{cases}$$

$$\begin{cases} 0 & v \le 0 \\ -8.57618 + 3.10509\,(21.0698 + 1.34645\,v)^{1/3} & v \le 122.784 \\ 9.16092 - 0.0427095\,(122.784 - v) & \text{True} \end{cases}$$





Comparing 15mL Falcon Tube models

We should use the fitted one, as we experimentally observed the other model predicting depths that were too large.

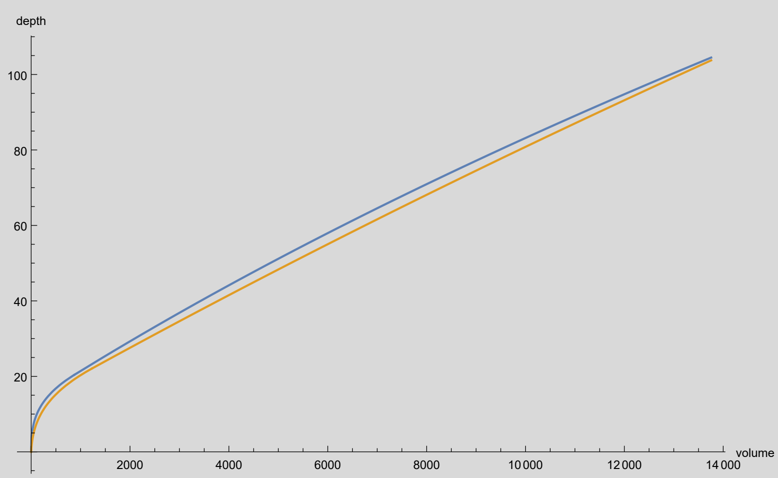
```
example1 = tubes[falcon15ml];
example2 = tubes[fittedFalcon15ml];
test @ example1;
test @ example2;
expr1 = depthFromVolume[example1, v]
expr2 = depthFromVolume[example2, v]
Plot[{expr1, expr2}, {v, 0, volume[example1]}, AxesLabel → {"volume", "depth"}, ImageSize → Large]
Plot[expr1 - expr2, {v, 0, volume[example1]}, AxesLabel → {"volume", "Δdepth"}, ImageSize → Large]
Show[ListPlot[{falconData}, AxesLabel → {"vol", "depth"}, PlotRange → All, AxesOrigin → {0, 0}, ImageSize → Large],
 Plot[{depthFromVolume[example1, v], depthFromVolume[example2, v]}, {v, 0, volume[example1]}]]
```

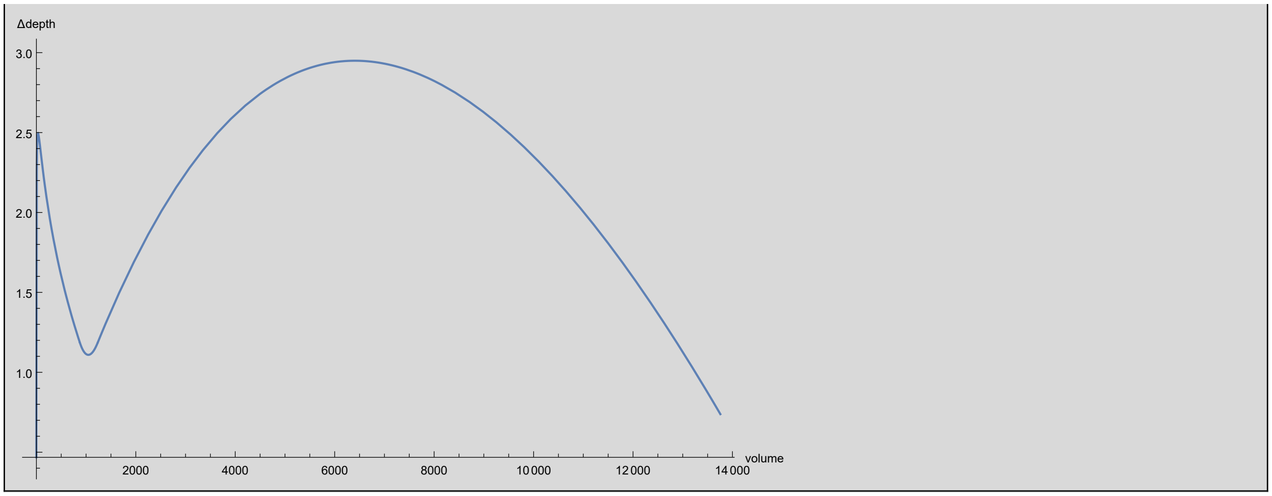
```
example1 →
 conicalTestTube[invertedFrustum[84.07, 7.665, 6.28], invertedFrustum[20.09, 6.28, 0.32], invertedSphericalCap[0.32, 0.32]]
```

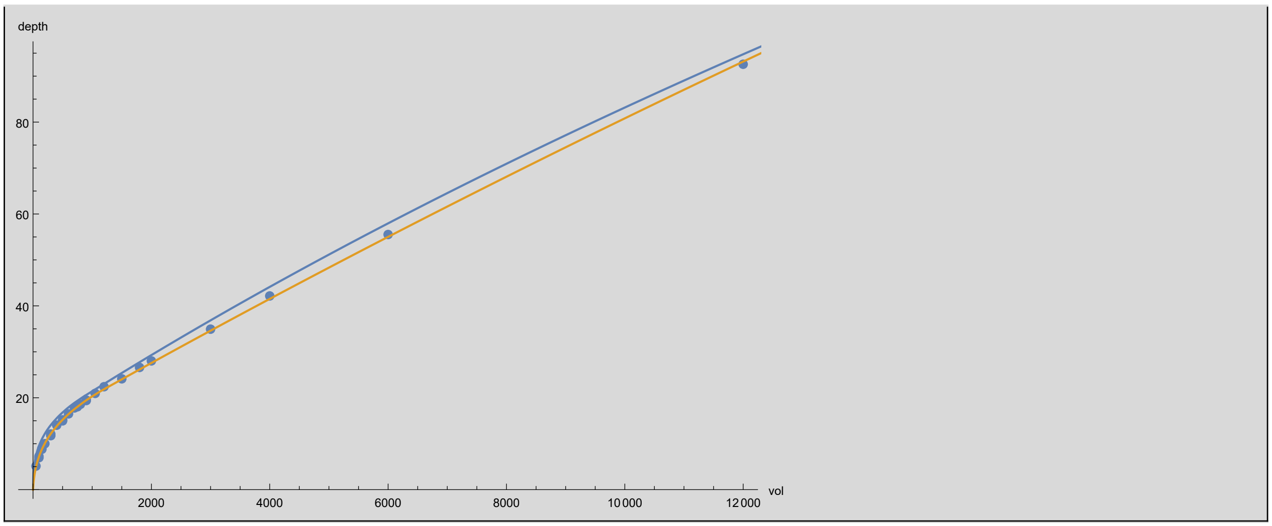
```
example2 → conicalTestTube[invertedFrustum[95.9755, 7.42952, 6.65602], invertedFrustum[22.0945, 6.65602, 1.14806], cylinder[0, 0]]
```

$$\begin{cases} 0.32 - \dfrac{0.0944778 - 0.16364\,i}{\left(0.205887 - 3\,v + \sqrt{3}\,\sqrt{-0.411775\,v + 3\,v^2}\right)^{1/3}} - \dfrac{\left(1 + i\,\sqrt{3}\right)\left(0.205887 - 3\,v + \sqrt{3}\,\sqrt{-0.411775\,v + 3\,v^2}\right)^{1/3}}{2\,(2\,\pi)^{1/3}} & v \le 0.0686291 \\ -0.758658 + 1.23996\,(0.267715 + 5.69138\,v)^{1/3} & v \le 874.146 \\ -360.788 + 13.8562\,(19665.7 + 1.32258\,v)^{1/3} & \text{True} \end{cases}$$

$$\begin{cases} 0 & v \le 0 \\ -4.60531 + 1.42955\,(33.4335 + 5.25971\,v)^{1/3} & v \le 1232.34 \\ -803.774 + 27.1004\,(27390.9 + 0.738644\,v)^{1/3} & \text{True} \end{cases}$$




Δdepth
3.0
2.5
2.0
1.5
1.0
2000
4000
6000
8000
10 000
12 000
14 000
volume


depth
80
60
40
20
2000
4000
6000
8000
10 000
12 000
vol