In [1]:
```
import sympy as sp
from sympy.plotting import (plot, plot_parametric)
```

VOLUME AND WORK IN PYTHON

This section is all about setting up the integration by hand, then doing the integration (and other steps as needed) in Python.

NOTE: The examples here (and in future overviews) are NOT a copy/paste to solve the problems in lab. However, they will USE many of the features you will use to solve your problems.

EXAMPLES:

1. Derive the formula for the volume of the frustrum of a cone with inner radius r, outer radius R, and height h.
2. The work required to stretch a spring from a natural length of 30cm to a length of 42cm is 3 Joules. Find the work required to stretch the spring an additional 12cm (from a length of 42cm to a length of 54cm).
3. A tank full of water is in the shape of a paraboloid (see a picture on p467 of the text) with a radius of 4 m and a height of 4m.

   a) Find the work done in pumping all the water out of the tank.

   b) Find the depth of water in the tank when 500000 J of work has been done.

RECALL the process we learned in the previous lab for solving these problems in Python: (a) List out the steps needed to solve the problem by hand, (b) find the Python commands needed to execute these steps.

For Example 1, we can use a solid of revolution to find the volume of the frustrum by rotating the line segment passing through the points (r,h) and (R,0) about the y-axis and adding that to the cylinder of radius r and height h (No Python skills needed for that-just a spatial understanding typically helpful for engineers!). So our process for solving this by hand would be:

1. Find the equation of the line (calculate m and use y = y0 + m * (x-x0))
2. Since we have a function of x rotating around the y-axis, we use cylindrical shells to set up our integral and use **integrate** in Python.

Note that r, R, and h are also symbolic parameters in this problem.

In [2]:
```
r,R,h,x=sp.symbols('r R h x')
# step 1: using delta-y over delta-x for slope and the first point for (x0,y0)
m=(0-h)/(R-r)
y=h+m*(x-r)
# step 2: integrate 2*sp.pi*x*f(x) (our equation) from x=r to x=R
Volume=sp.integrate(2*sp.pi*x*y,(x,r,R))+sp.pi*r**2*h  # rotated volume plus cylinder volume
print('The volume of our frustrum is',Volume.simplify())  #simplifying to make it look nicer
```

```
The volume of our frustrum is pi*h*(R**2 + R*r + r**2)/3
```

Notice that if r=0 we have the entire cone and the formula for volume is the familiar V = 1/3 * pi * R^2 * h.

For Example 2, we use Hooke's Law, F = kx. The steps to solve by hand are as follows:

1. Integrate F(x) = k * x from x=0 to x=0.12 to find the work in terms of k (Python command **sp.integrate**)
2. Solve Work = 3 for k (Python command **sp.solve**)
3. Substitute this value of k into the Force (Python command **subs**)
4. Integrate this new Force from x=.12 to x=.24 (Python command **sp.integrate**)

In [3]:
```
k,x=sp.symbols('k x')
# Step 1
F=k*x
Work1=sp.integrate(F,(x,0,0.12))
# Step 2
eqn=Work1-3
print(sp.solve(eqn,k)) #Ran code here to see how many solutions we got and which are valid ones.  Only one answer.
knew=sp.solve(eqn,k)[0]  #REMEMBER-the first element in the list is element number ZERO!!!!
# Step 3
Fnew=F.subs(k,knew)
print('The new force equation is F=',Fnew)
# You don't have to enter the line above, but showing output after each step helps with debugging when needed!
# Step 4
Work2=sp.integrate(Fnew,(x,0.12,0.24))
print('The work required is',Work2,'Joules.')
```

```
[416.666666666667]
The new force equation is F= 416.666666666667*x
The work required is 9.00000000000000 Joules.
```

For question 3, we have to set up the integral required for the work (which you can do entirely by hand). If we place a coordinate system with the origin at the vertex of the paraboloid, the slices are circular disks with volume pi * x^2 * dy. Putting x in terms of y, we notice the upper right "corner" of the paraboloid has coordinates (4,4). Since the outline is the parabola y = ax^2, we find a = 1/4, so y = 1/4x^2 (or x^2 = 4y). Of course, we still have to multiply this volume by density (1000)(9.8) and by the distance traveled (4-y the way we set up the coordinate system). If this doesn't make sense, spend some time studying Section 6.4!

Once we have set up the integral to find work, the steps to solve the rest of the problem are as follows:

Part a: **sp.integrate** from y=0 to y=4 the integral you found by hand to obtain work.

Part b

Step 1: **sp.integrate** from y=h to y=4 the integral you found by hand to obtain work.

Step 2: **sp.solve** the work = 500000 for h (where "work" refers to the result from Step 1).

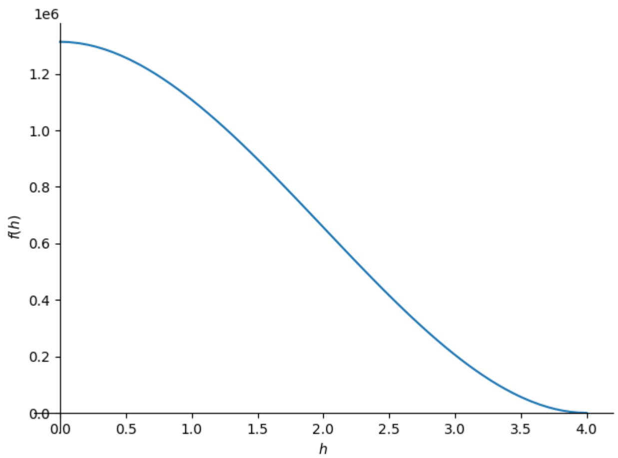
In [4]:
```
# Part a
y=sp.symbols('y')
Fslice=9800*sp.pi*4*y
Wslice=Fslice*(4-y)
Work=sp.integrate(Wslice,(y,0,4))
print('The work required to pump all the water out of the tank is',Work,'Joules')
# Part b (Step 1)
h=sp.symbols('h')
Work=sp.integrate(Wslice,(y,h,4))
print('The work required to pump to a depth of h is',Work)
hsol=sp.solve(Work-500000,h)
print(hsol)
```

```
The work required to pump all the water out of the tank is 1254400*pi/3 Joules
The work required to pump to a depth of h is 39200*pi*h**3/3 - 78400*pi*h**2 + 1254400*pi/3
[2 - 12/((-1/2 - sqrt(3)*I/2)*(-216 + 27*(-1875 + 1568*pi)/(98*pi) + sqrt(-186624 + (-432 + 27*(-1875 + 1568*pi)/(49*pi))**2)/2)**(1/3)) - (-1/2 - sqrt(3)*I/2)*(-216 + 27*(-1875 + 156
8*pi)/(98*pi) + sqrt(-186624 + (-432 + 27*(-1875 + 1568*pi)/(49*pi))**2)/2)**(1/3)/3, 2 - (-1/2 + sqrt(3)*I/2)*(-216 + 27*(-1875 + 1568*pi)/(98*pi) + sqrt(-186624 + (-432 + 27*(-1875
+ 1568*pi)/(49*pi))**2)/2)**(1/3)/3 - 12/((-1/2 + sqrt(3)*I/2)*(-216 + 27*(-1875 + 1568*pi)/(98*pi) + sqrt(-186624 + (-432 + 27*(-1875 + 1568*pi)/(49*pi))**2)/2)**(1/3)), 2 - (-216 +
27*(-1875 + 1568*pi)/(98*pi) + sqrt(-186624 + (-432 + 27*(-1875 + 1568*pi)/(49*pi))**2)/2)**(1/3)/3 - 12/(-216 + 27*(-1875 + 1568*pi)/(98*pi) + sqrt(-186624 + (-432 + 27*(-1875 + 1568
*pi)/(49*pi))**2)/2)**(1/3)]
```

Answers look too complicated. We could try restricting h to be a real symbolic variable (**h=symbols('h',real=True)**) but even this doesn't help. So let's try plotting and using **nsolve** as mentioned back in Lab 1.

In [5]:
```
matplotlib notebook
```

In [6]:
```
plot(Work,(h,0,4)) # solution appears to be about 2.5
hsol=sp.nsolve(Work-500000,h,2.5)
print('The depth of water in the tank is approximately',hsol,'meters.')
```




```
The depth of water in the tank is approximately 2.32107278576941 meters.
```

In [ ]: