# Problems list 4

Jonas Noack

17. Let C be a right circular cone with height h, semi-aperture α, and apex at the origin, lying on the plane z = 0 (i.e., tangent to z = 0), enclosed in the first octant, and such that the line segment connecting its apex with the center point of its basis orthogonally projects onto z = 0 in the line x = y, z = 0. Obtain a parametrization and the cartesian equation of the cone in the standard coordinate system.

The solution can be obtained using the following SageMath script:

```
from sage.plot.plot3d.transform import rotate_arbitrary

s = var('s')
t = var('t')


# angle of the cone
angle = pi/8
# height of the cone
h = 3

# cone positioned in origin with angle defined above
cone_function(s,t) = (s * sin(angle) * cos(t),s * sin(angle) *
sin(t), s * cos(angle))
# render for defined height
cone = parametric_plot3d(cone_function, (t, 0, 2 * pi), (s, 0, h),
opacity=0.4)

cone_center_funciton(s) = (0,0,s)
cone_center = parametric_plot(cone_center_funciton, (s,0,h),
color="red")

# we define a line so that we rotate around to achieve the needed
position:
line_function(s) = (s,-s,0)
line = parametric_plot3d(line_function, (s, -1, 1), color="red")

# create a rotation matrix for this line with the angle of the cone
rotation = rotate_arbitrary((1,-1,0), (pi/2) - angle)
print("rotation matrix:")
print(rotation)
```

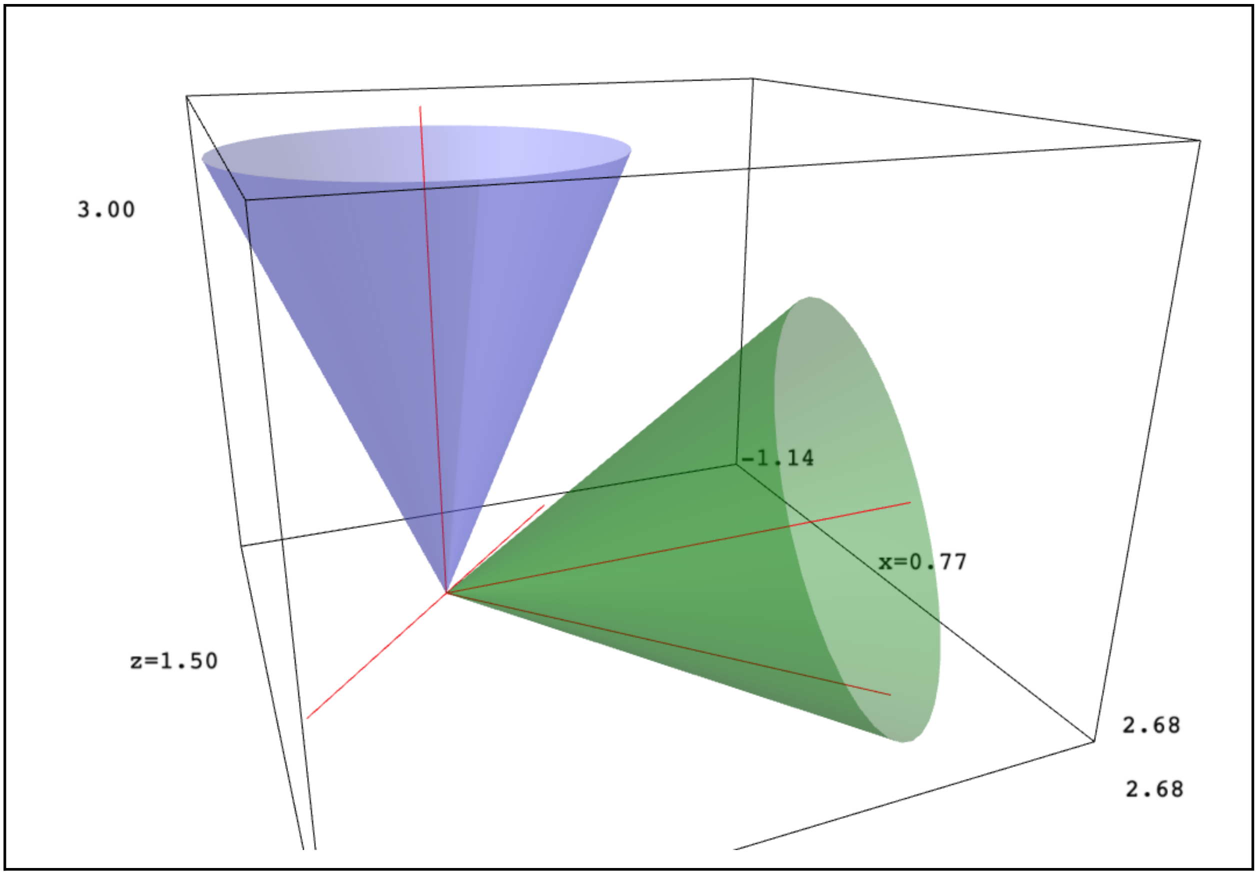
```
# apply the rotation to the cone
rotated_cone_function = rotation * vector(cone_function)
rotated_cone = parametric_plot3d(rotated_cone_function, (t, 0, 2 *
pi), (s, 0, h), color='green', opacity=0.4)

# show the center and its projection
rotated_cone_center_funciton = rotation *
vector(cone_center_funciton)
rotated_cone_center = parametric_plot(rotated_cone_center_funciton,
(s,0,h), color="red")
rotated_cone_center_projected_on_ground =
parametric_plot((rotated_cone_center_funciton[0],
rotated_cone_center_funciton[1], 0), (s,0,h), color="red")

cone + cone_center + line + rotated_cone + rotated_cone_center +
rotated_cone_center_projected_on_ground
```

Output:

```
rotation matrix:
[ 0.6913417161825448  -0.308658283817455  0.6532814824381882]
[ -0.308658283817455  0.6913417161825448  0.6532814824381882]
[-0.6532814824381882 -0.6532814824381882 0.38268343236508984]
```




It can also be solved using this Geogebra setup: