Glass Slit

This shows a small improvement in seeing for a random collection of telescopes f/ratios and diameters.

See the bottom for the output.

In [37]:
```
import numpy as np
import pandas as pd
air = 1.0003
bk7 = 1.5300
thickness = 2   # milimeters
seeing = 2.0 # [arcseconds]
```

```
Add a few functions for focal length from f/ration and diameter and
to convert a proposed seeing disc in arcseconds to a size at the
focal plane.
```

In [35]:
```
def focallen(fratio,diameter):
    """Convert diameters [inches] into focal length [mm]"""
    return diameter*25.4 * fratio

def seeingdisc(seeing_arcseconds, flmm):
    """Convert a seeing disc in arcseconds into a disc size at foca
    flmm in mm away into a disc xxx microns in diameter"""
    return flmm * 1000.0 * seeing_arcseconds/206264.8

```

```
The controlling constants.
```

In [38]:
```
fratios    = np.array([11.0, 10.0, 8.0, 7.0, 6.7,5.0, 4.0])
diameters  = np.array([11.0,10.0,6.0,6.0,14.5,10.0,24.0])
focallens  = focallen(fratios,diameters)
angles     = np.arctan(1.0/fratios)   # radians
halfangles = np.sqrt(1.0 - np.cos(angles/2.0))
discs      = seeingdisc(seeing,focallens)
```

```
A few preliminary definitions
```

In [39]:
```
data       = list(zip(fratios,diameters,focallens,angles,halfangles
df         = pd.DataFrame(data,columns=["f/ratios","diameter [in]",
print(df)
```

```
   f/ratios  diameter [in]  fl [mm]  angles [rad]  halfangles [rad]   d
isc[um]
0      11.0           11.0  3073.40      0.090660          0.032050  2
9.800528
1      10.0           10.0  2540.00      0.099669          0.035235  2
4.628536
2       8.0            6.0  1219.20      0.124355          0.043959  1
1.821697
3       7.0            6.0  1066.80      0.141897          0.050158  1
0.343985
4       6.7           14.5  2467.61      0.148160          0.052371  2
3.926622
5       5.0           10.0  1270.00      0.197396          0.069762  1
2.314268
6       4.0           24.0  2438.40      0.244979          0.086559  2
3.643394
```

Snells Law

Apply

$$\frac{n_1}{n_2} = \frac{\theta_2}{\theta_1}$$

note the angles change indices.

In [25]:
```
def refraction(n1,n2,angles):
    """return array of corresponding output angles"""
    return np.arcsin(n1/n2 * np.sin(angles))

```

Intermediate Conclusion

Here a slight improvement in the seeing disc is noted.

Note

The full import of diffraction in a converting beam has not been fully factored in. A bit more research to do.

In [46]:

```
1  deflections = refraction(air,bk7,angles)
2  newdisc = discs - 2.0 * thickness * deflections
3  pd2 = pd.DataFrame(list(zip(fratios,focallens,discs,newdisc)), colu
4  print(pd2)
```

```
   f/ratios  fl [mm]  Seeing Disc       disc
0      11.0  3073.40    29.800528  29.563624
1      10.0  2540.00    24.628536  24.368133
2       8.0  1219.20    11.821697  11.496969
3       7.0  1066.80    10.343985   9.973616
4       6.7  2467.61    23.926622  23.539974
5       5.0  1270.00    12.314268  11.799976
6       4.0  2438.40    23.643394  23.006436
```