nnratio

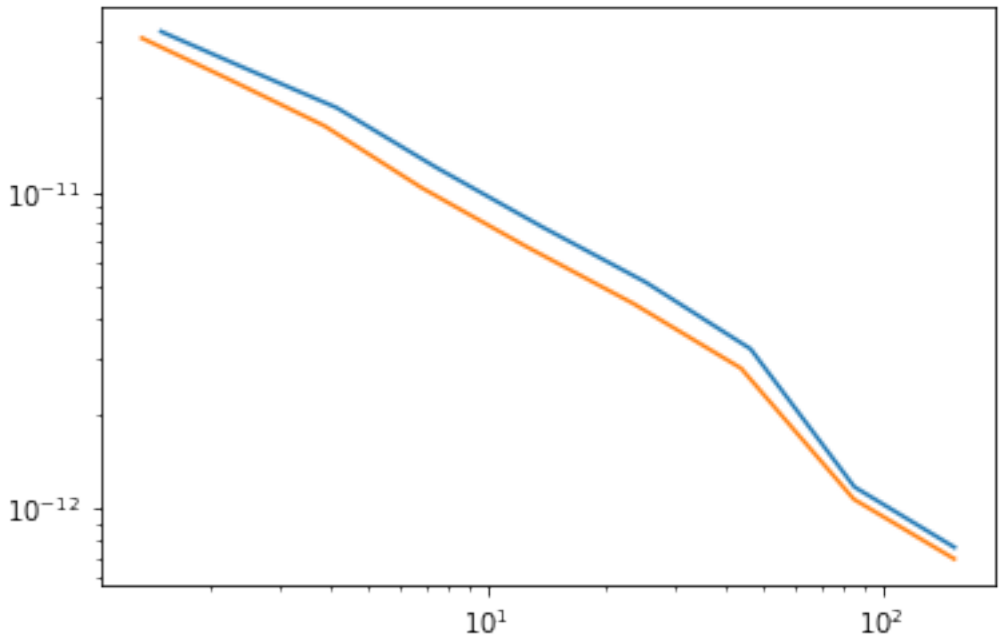
January 28, 2019

In [13]:
```
import numpy as np
import matplotlib.pyplot as plt
import scipy as sp
from numpy.polynomial.polynomial import Polynomial
from scipy.interpolate import interp1d
from scipy.interpolate import lagrange
%matplotlib inline
```

In [3]:
```
#import data from plot digitizer
nnafx = np.array([1.4966542,2.496218,4.1633563,7.46208,13.865235,25.21241,46.508884,85
nnb4y = np.array([3.09E-11,2.27E-11,1.63E-11,1.04E-11,6.75E-12,4.46E-12,2.78E-12,1.07E
nnafy = np.array([3.24E-11,2.45E-11,1.86E-11,1.20E-11,7.77E-12,5.21E-12,3.20E-12,1.17E
nnb4x = np.array([1.3380892,2.2522328,3.8739629,6.808623,12.588559,23.444796,43.976784
```

In [4]:
```
#make sure data resembles original copy
plt.loglog(nnafx,nnafy)
plt.loglog(nnb4x,nnb4y)
```

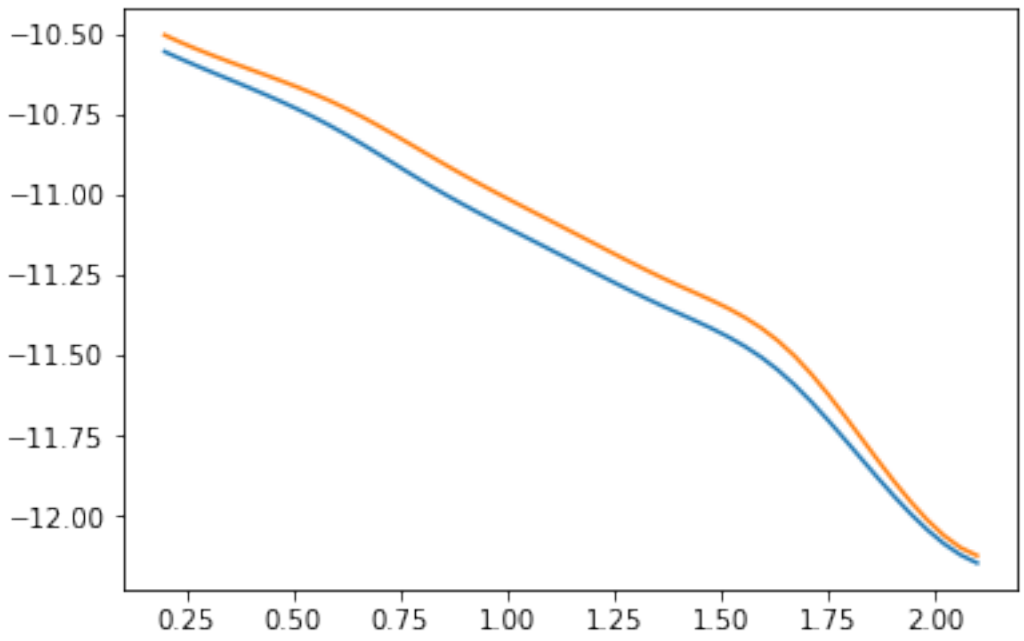
Out[4]:
```
[<matplotlib.lines.Line2D at 0x2178339f748>]
```



In [60]:
```
#move to cartesian space to make good fit
xb4 = np.log10(nnb4x)
yb4 = np.log10(nnb4y)
xaf = np.log10(nnafx)
yaf = np.log10(nnafy)
x = np.linspace(.2,2.1,50)
```

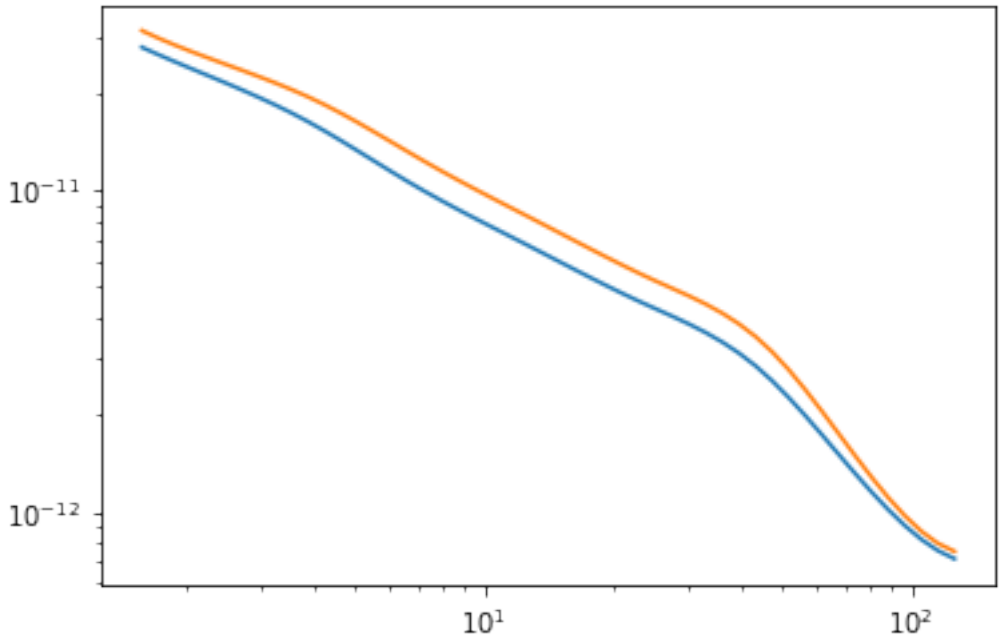
In [61]:
```
#use cubic interpolation, the shape of the graph looks like this will work best
fb4 = interp1d(xb4, yb4,kind='cubic')
plt.plot(x,fb4(x))
faf = interp1d(xaf, yaf,kind='cubic')
plt.plot(x,faf(x))
```

Out[61]:
```
[<matplotlib.lines.Line2D at 0x21786fe9898>]
```



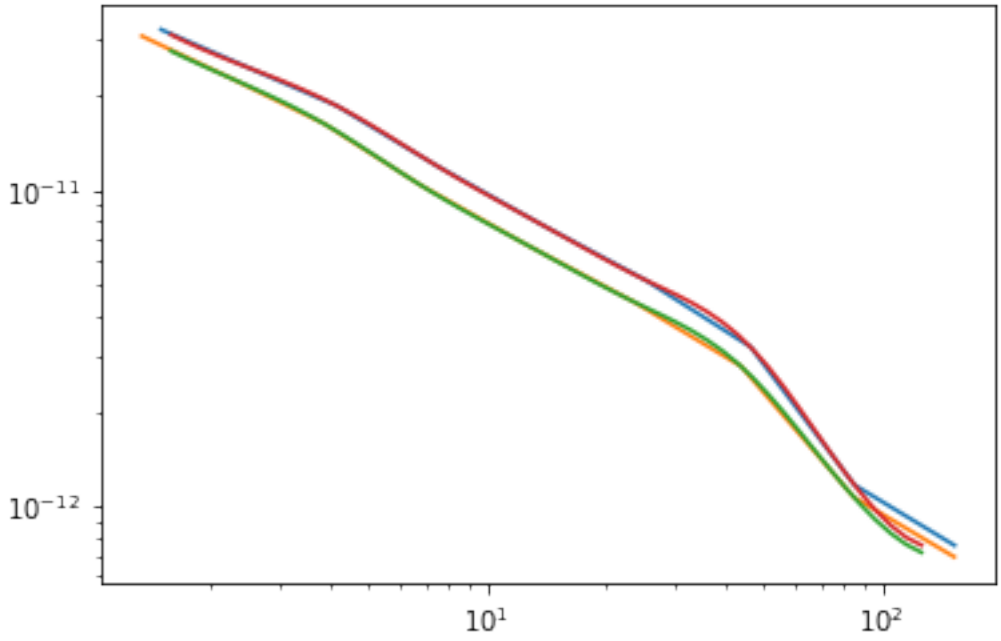
In [72]:
```
#move fit to log space
x = np.linspace(.2,2.1,50)
fb4log = 10**fb4(x)
faflog = 10**faf(x)
x = 10**x
plt.loglog(x,fb4log)
plt.loglog(x,faflog)
```

Out[72]:
```
[<matplotlib.lines.Line2D at 0x2178746fba8>]
```



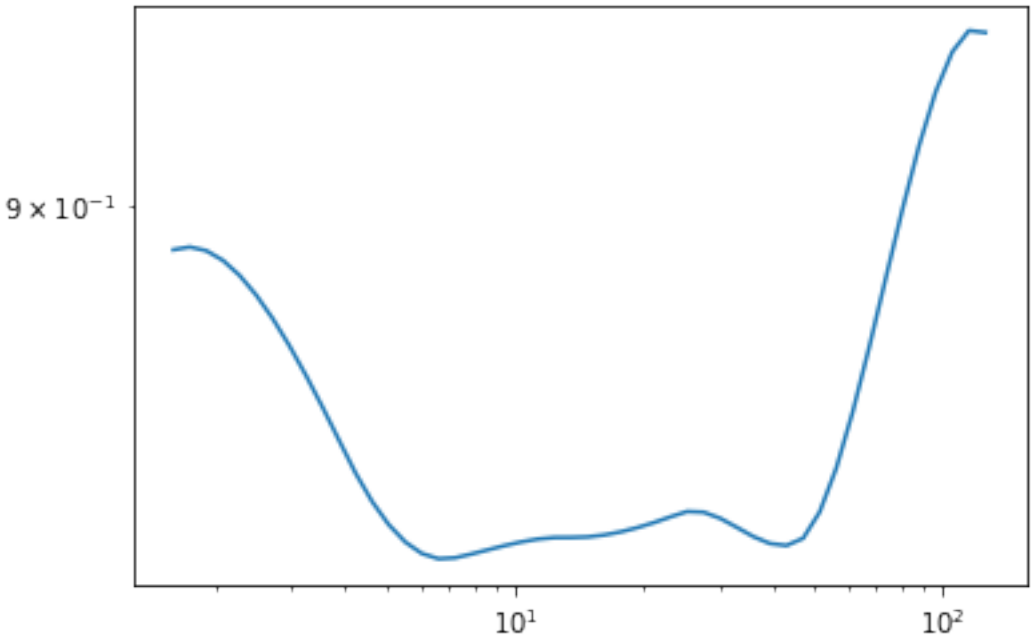
In [63]:
```
#make sure fits lie on top of data
plt.loglog(nnafx,nnafy)
plt.loglog(nnb4x,nnb4y)
plt.loglog(x,fb4log)
plt.loglog(x,faflog)
```

Out[63]: [<matplotlib.lines.Line2D at 0x2178721c3c8>]



In [78]:
```
#plot ratio
plt.loglog(x,fb4log/faflog)
```

Out[78]:
```
[<matplotlib.lines.Line2D at 0x217865a3588>]
```



In [77]:
```
#compute ratio
np.average(fb4log)/np.average(faflog)
```

Out[77]:
```
0.8460958038078833
```

In [ ]: