# Kerr Effect

April 22, 2019

In [2]:
```
from math import*
import numpy as np
import matplotlib.pyplot as plt
import scipy.special as sp

from matplotlib import pylab

from numpy import arange,array,ones
from scipy import stats
from scipy import optimize
```

In [3]:
```
U = [14,300,325,350,375,400,425,450,475,500,525,550,575,600,625,650,675,700]
I = [.029,.055,.060,.065,.074,.090,.110,.125,.141,.164,.169,.175,.141,.120,.086,.070,.07
I_o = 0.194
rel_I = [(x/I_o) for x in I]
delta = [(360/np.pi)*np.arcsin(sqrt(x)) for x in rel_I]
```

In [4]:
```
delta[12] = 360 - delta[12]
delta[13] = 360 - delta[13]
delta[14] = 360 - delta[14]
delta[15] = 360 - delta[15]
delta[16] = 360 + delta[16]
delta[17] = 360 + delta[17]
```

In [5]:
```
Usqr = [u**2 for u in U]
```

In [6]:
```
slope, intercept, r_value, p_value, std_err = stats.linregress(delta,Usqr)
```

In [7]:
```
slope
```

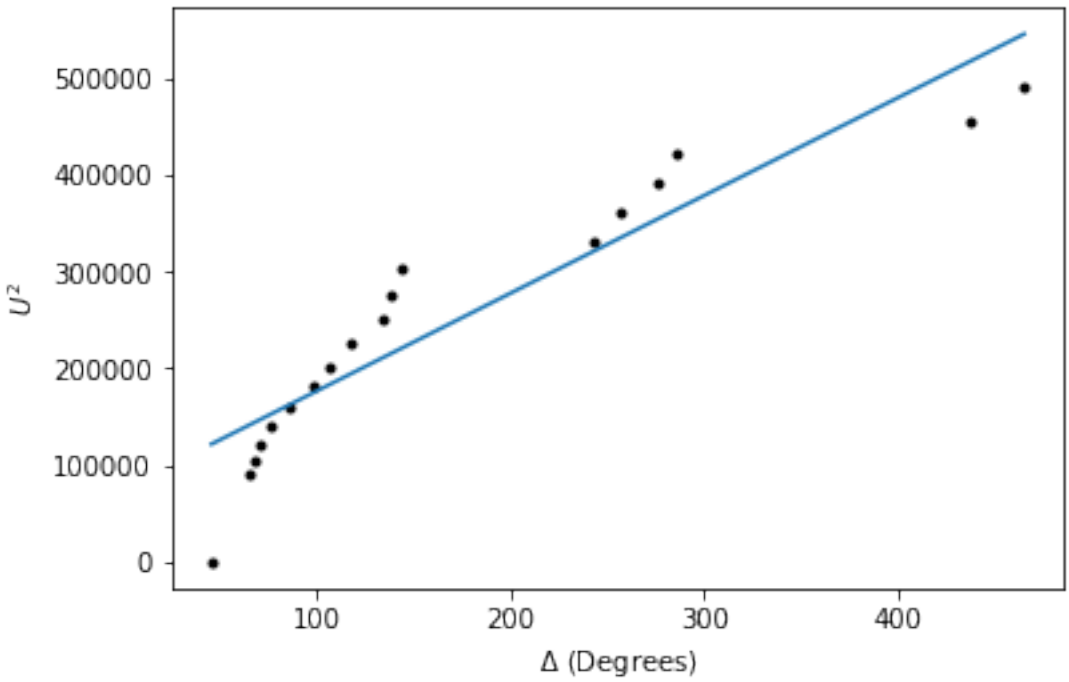
Out[7]:
```
1007.8946121033131
```

In [8]:
```
intercept
```

Out[8]:
```
76005.9884206331
```

In [9]:
```
plt.plot(delta, Usqr, '.k');
plt.ylabel(r'$U^2$');
plt.xlabel(r'$\Delta$ (Degrees)');

fit = [(slope*x + intercept) for x in delta]
plt.plot(delta,fit);
```



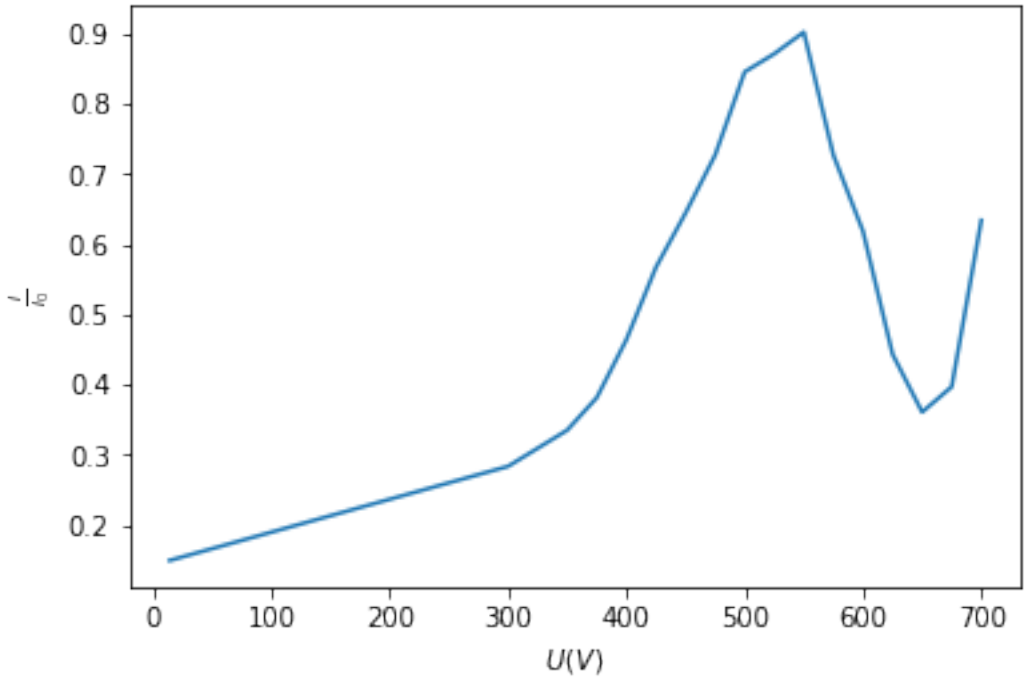

```
In [12]: slope

Out[12]: 1007.8946121033131

In [11]: def test_func(x, a, b, c):
             return a * np.sin(b * x**2)**2 + c

         params, params_covariance = optimize.curve_fit(test_func, U, rel_I, p0=[0.6,0.000009,0.

         plt.plot(U,rel_I);
         plt.xlabel(r'$U (V)$');
         plt.ylabel(r'$\frac{I}{I_0}$');
```

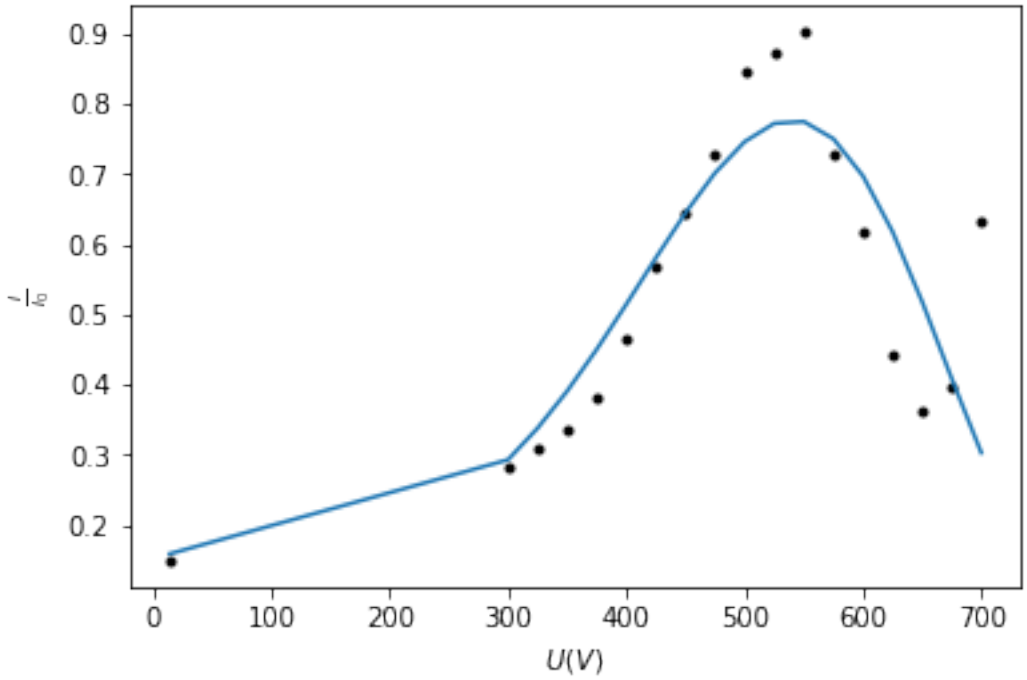



```
In [13]: params[0]

Out[13]: 0.6178369978101451

In [14]: fitt = [test_func(x,params[0],params[1],params[2]) for x in U]

In [18]: plt.plot(U,rel_I,'.k');
         plt.xlabel(r'$U (V)$');
         plt.ylabel(r'$\frac{I}{I_0}$');
         plt.plot(U,fitt);
```


0.9
0.8
0.7
0.6
0.5
0.4
0.3
0.2
$\frac{I}{I_0}$
0
100
200
300
400
500
600
700
$U(V)$