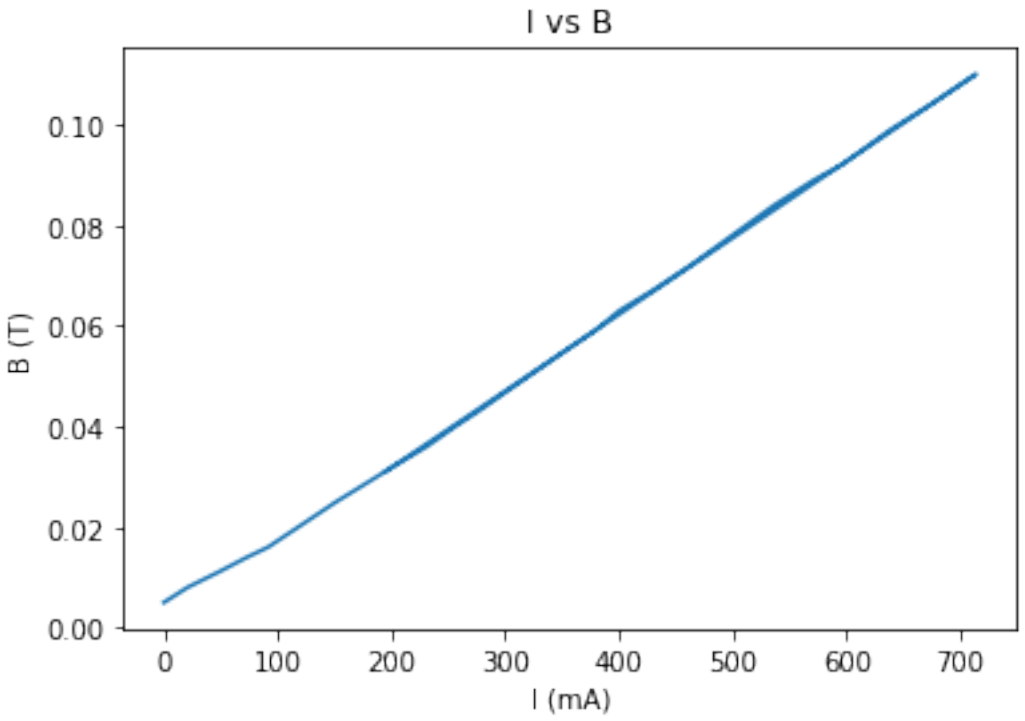
Beta decay

April 2, 2020

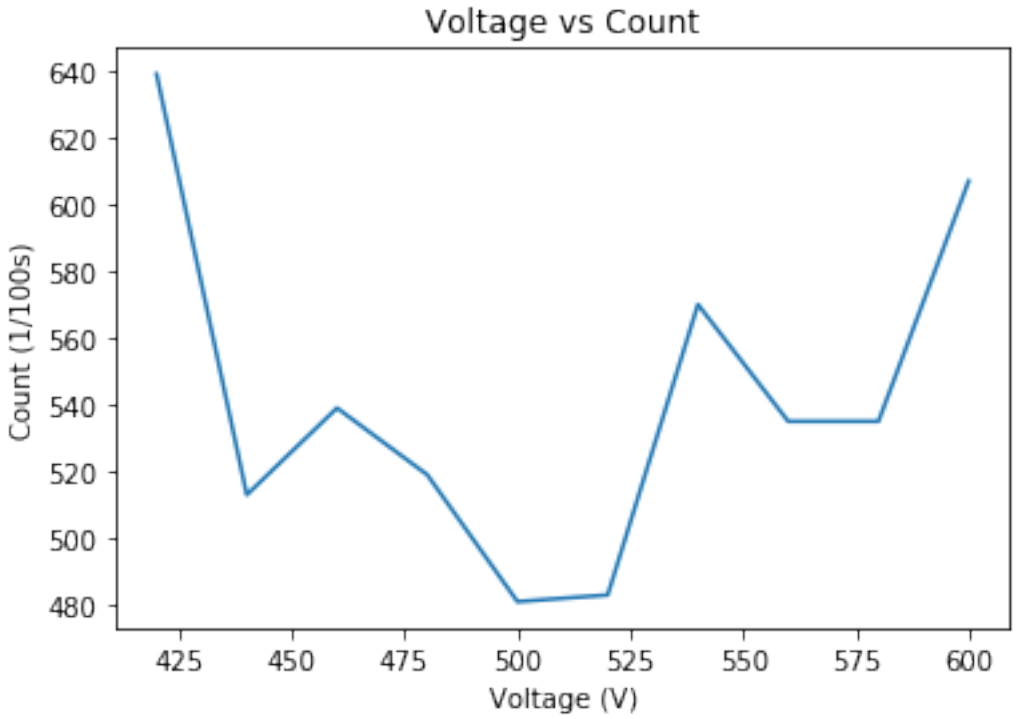
```
In [295]: import matplotlib.pyplot as plt
          import numpy as np
          from IPython.display import Image
          from scipy.stats import linregress
```

```
In [298]: I = [0, 20.8, 47.6, 73.1, 91.6, 151.3, 180.1, 201.4, 231, 256.2, 276.4, 308, 359.1, 37
               429.4, 462.7, 481.2, 536.5, 571.4, 595.8, 638.7, 674.1, 713.6, 194.6]
          T = [.005, .008, .011, .014, .016, .025, .029, .032, .036, .04, .043, .048, .056, .059
              .075, .084, .089, .092, .099, .104, .110, .031]
          plt.plot(I, T)
          plt.xlabel('I (mA)')
          plt.ylabel('B (T)')
          plt.title('I vs B')
          plt.show()
          print(linregress(I, T))
```



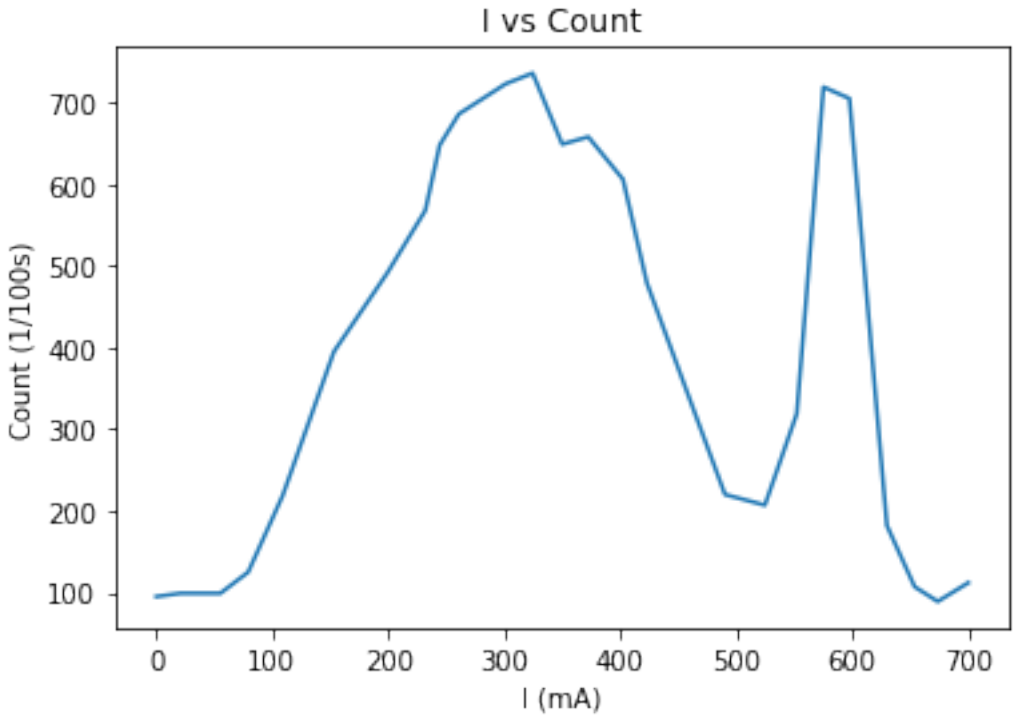
```
LinregressResult(slope=0.00014959722084772217, intercept=0.002812295732654832, rvalue=0.99957375
```

```
In [299]: V = [420, 440, 460, 480, 500, 520, 540, 560, 580, 600]
          I = []
          count = [639, 513, 539, 519, 481, 483, 570, 535, 535, 607]
          count_error = []
          for x in count:
              count_error.append(np.sqrt(x))
          plt.plot(V, count)
          plt.xlabel('Voltage (V)')
          plt.ylabel('Count (1/100s)')
          plt.title('Voltage vs Count')
          plt.show()
```




```
In [300]: I = [0, 20.9, 55.3, 79.3, 109.1, 153.2, 199.5, 232, 244.4, 260.9, 301.3, 324.5, 350, 3
              524.7, 552.2, 575.2, 597.6, 629.9, 653.8, 673.7, 700.1]
          count = [95, 99, 99, 125, 219, 395, 492, 568, 648, 686, 723, 736, 649, 658, 606, 479,
                   182, 107, 89, 112]
          count_error = []
          for x in count:
```

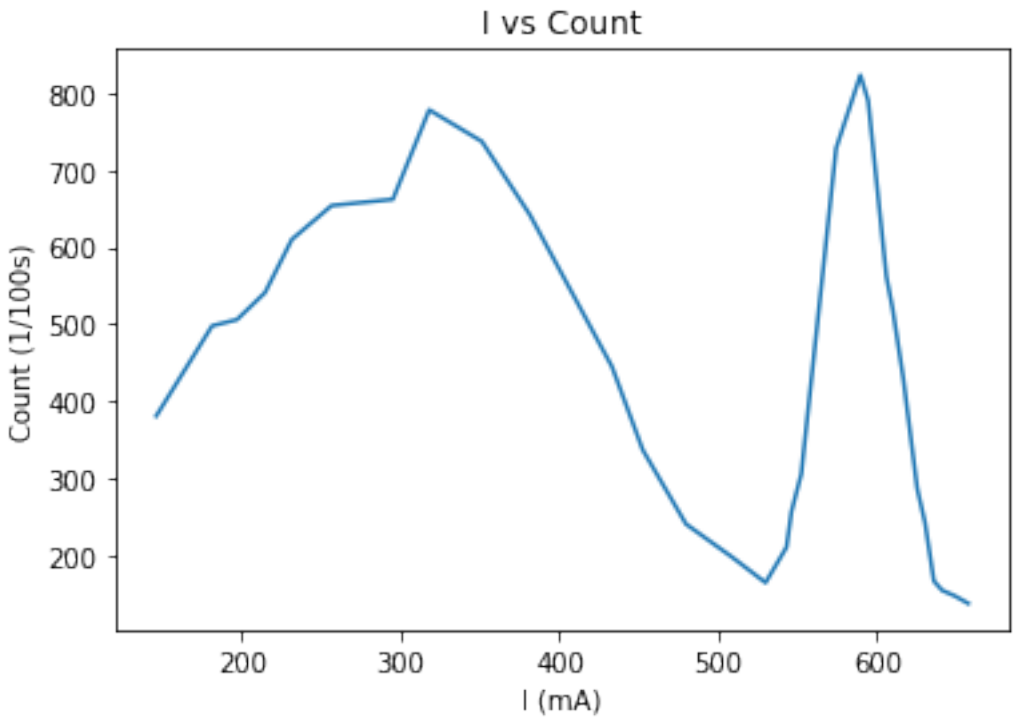
```
        y = np.sqrt(x)
        y = round(y,2)
        count_error.append(y)
    plt.plot(I, count)
    plt.xlabel('I (mA)')
    plt.ylabel('Count (1/100s)')
    plt.title('I vs Count')
    plt.show()
    print("I(mA)\t| Count(1/100s)")
    print("_____________________\n")
    for x in range(len(I)):
        print(I[x],"\t|",str(count[x])+" +- "+str(count_error[x]))
```




```
I(mA)        | Count(1/100s)
---------------------

0          | 95 +- 9.75
20.9         | 99 +- 9.95
55.3         | 99 +- 9.95
79.3         | 125 +- 11.18
109.1         | 219 +- 14.8
153.2         | 395 +- 19.87
199.5         | 492 +- 22.18
232          | 568 +- 23.83
```

```
244.4         | 648 +- 25.46
260.9         | 686 +- 26.19
301.3         | 723 +- 26.89
324.5         | 736 +- 27.13
350         | 649 +- 25.48
372.5         | 658 +- 25.65
402.4         | 606 +- 24.62
423         | 479 +- 21.89
464         | 321 +- 17.92
490.3         | 220 +- 14.83
524.7         | 207 +- 14.39
552.2         | 319 +- 17.86
575.2         | 719 +- 26.81
597.6         | 705 +- 26.55
629.9         | 182 +- 13.49
653.8         | 107 +- 10.34
673.7         | 89 +- 9.43
700.1         | 112 +- 10.58
```

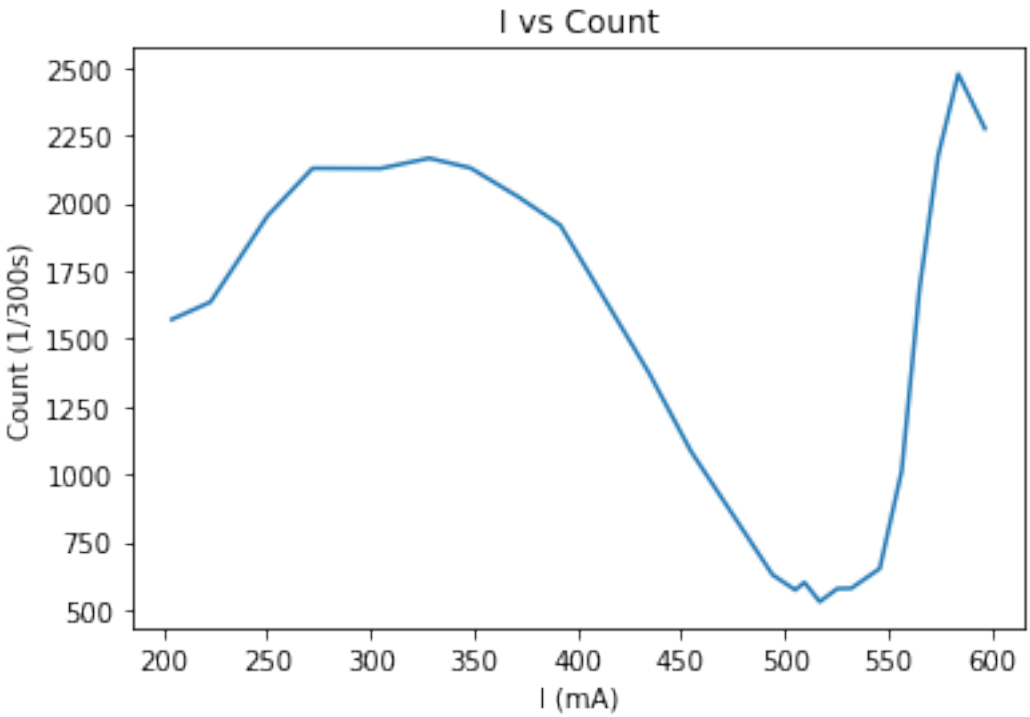
```
In [301]: I = [146, 181.2, 196.8, 214.5, 231.4, 256.5, 295, 317.9, 351.1, 381.3, 433, 452.4, 479
              552.2, 574.1, 589.4, 594.3, 605.8, 609.8, 616.5, 625.2, 630.1, 635.7, 640.9, 648.7
          count = [381, 498, 506, 541, 610, 654, 662, 778, 737, 642, 445, 337, 241, 198, 165, 21
                   790, 562, 519, 429, 287, 243, 167, 155, 148, 138]
          count_error = []
          for x in count:
              y = np.sqrt(x)
              y = round(y,2)
              count_error.append(y)
          plt.plot(I, count)
          plt.xlabel('I (mA)')
          plt.ylabel('Count (1/100s)')
          plt.title('I vs Count')
          plt.show()
          print("I(mA)\t| Count(1/100s)")
          print("_____________________\n")
          for x in range(len(I)):
              print(I[x],"\t|",str(count[x])+" +- "+str(count_error[x]))
```



I(mA)	Count(1/100s)
146	381 +- 19.52
181.2	498 +- 22.32
196.8	506 +- 22.49
214.5	541 +- 23.26
231.4	610 +- 24.7
256.5	654 +- 25.57
295	662 +- 25.73
317.9	778 +- 27.89
351.1	737 +- 27.15
381.3	642 +- 25.34
433	445 +- 21.1
452.4	337 +- 18.36
479.6	241 +- 15.52
508.7	198 +- 14.07
529.7	165 +- 12.85
542.9	211 +- 14.53
546	257 +- 16.03
552.2	306 +- 17.49
574.1	728 +- 26.98
589.4	823 +- 28.69
594.3	790 +- 28.11

```
605.8          | 562 +- 23.71
609.8          | 519 +- 22.78
616.5          | 429 +- 20.71
625.2          | 287 +- 16.94
630.1          | 243 +- 15.59
635.7          | 167 +- 12.92
640.9          | 155 +- 12.45
648.7          | 148 +- 12.17
657.4          | 138 +- 11.75
```

```
In [302]: I = [204, 222.7, 250.3, 272, 304.8, 328.4, 348.3, 371.3, 391.7, 434.5, 454.8, 494.2, 5
               546, 556.6, 565.2, 574.2, 583.9, 596.8]
          count = [1571, 1635, 1953, 2128, 2127, 2165, 2129, 2023, 1918, 1374, 1085, 630, 574, 6
                   1684, 2174, 2475, 2274]
          count_error = []
          for x in count:
              y = np.sqrt(x)
              y = round(y,2)
              count_error.append(y)
          plt.plot(I, count)
          plt.xlabel('I (mA)')
          plt.ylabel('Count (1/300s)')
          plt.title('I vs Count')
          plt.show()
```



```
In [303]: print("I(mA)\t| Count(1/300s)")
          print("_____________________\n")
          for x in range(len(I)):
              print(I[x],"\t|",str(count[x])+" +- "+str(count_error[x]))
          newI = []
          for y in I:
              newI.append(y*0.00014959722084772217)
```

```
I(mA)        | Count(1/300s)
---------------------

204          | 1571 +- 39.64
222.7          | 1635 +- 40.44
250.3          | 1953 +- 44.19
272          | 2128 +- 46.13
304.8          | 2127 +- 46.12
328.4          | 2165 +- 46.53
348.3          | 2129 +- 46.14
371.3          | 2023 +- 44.98
391.7          | 1918 +- 43.79
434.5          | 1374 +- 37.07
454.8          | 1085 +- 32.94
494.2          | 630 +- 25.1
505.2          | 574 +- 23.96
509.6          | 602 +- 24.54
517          | 531 +- 23.04
525.5          | 579 +- 24.06
532.2          | 580 +- 24.08
546          | 653 +- 25.55
556.6          | 1013 +- 31.83
565.2          | 1684 +- 41.04
574.2          | 2174 +- 46.63
583.9          | 2475 +- 49.75
596.8          | 2274 +- 47.69
```

```
In [308]: #step 10
          # calculating pc/e for each value of I
          dp_p = 0.1
          r = 0.0381
          c = 2.9979*10**8
          pc_e = []
          print("I(mA)\t| pc/e")
          print("_____________________\n")
          for N in range(len(I)):
              pc_e.append(newI[N]*0.0381*c)
              print(I[N],"\t|",int(pc_e[N]))
```

```
I(mA)        | pc/e
---------------------

204          | 348574
222.7          | 380527
250.3          | 427687
272          | 464766
304.8          | 520811
328.4          | 561136
348.3          | 595139
371.3          | 634440
391.7          | 669297
434.5          | 742429
454.8          | 777116
494.2          | 844439
505.2          | 863234
509.6          | 870753
517          | 883397
525.5          | 897921
532.2          | 909369
546          | 932949
556.6          | 951062
565.2          | 965756
574.2          | 981135
583.9          | 997709
596.8          | 1019751
```

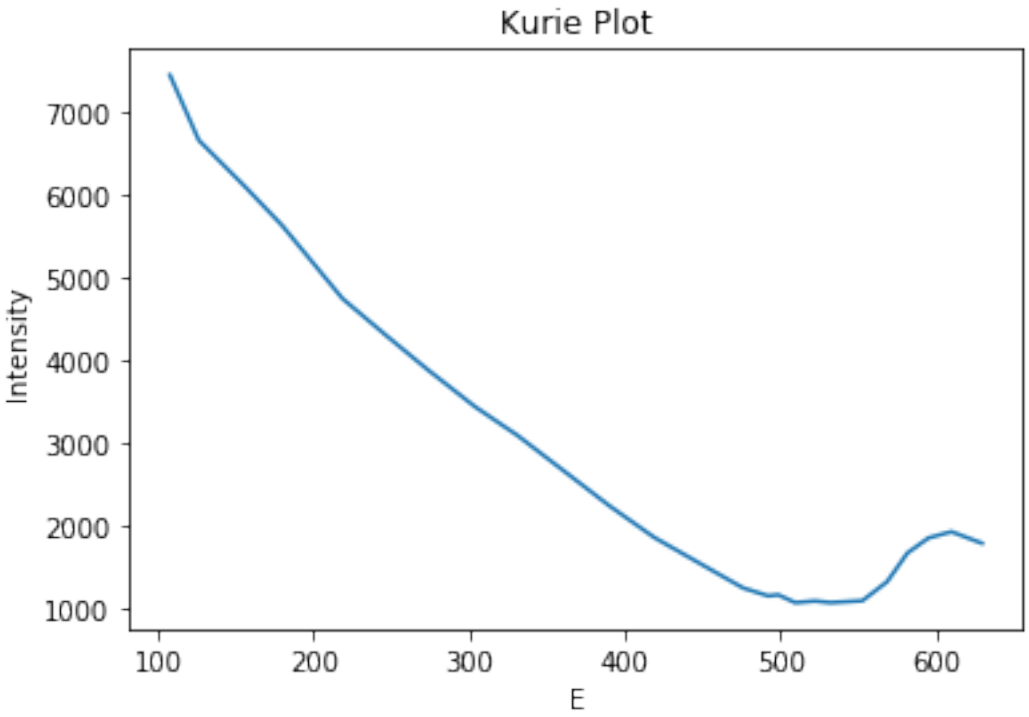
In [354]:
```
#step 11
E_0 = 511003.4
print("The conversion energy is 624 keV. This is lower than the gamma energy 662 keV b
print("\nThe max value of the energy is",(np.sqrt((E_0**2) + max(pc_e)**2) - E_0)/1000
print("\nThe higest value in pc_e is",int(max(pc_e)))
```

```
The conversion energy is 624 keV. This is lower than the gamma energy 662 keV because
the energy released after conversion is the amount of energy
minus the binding energy, since that is lost when the electron is released.
The accepted value of the conversion energy is 624 keV. Using our data and the formula
provided, we calculated a conversion energy of 654.5 using a Brc of 1013.46 T*m^2/s.

The max value of the energy is 723.065053656

The higest value in pc_e is 1123298
```

In [334]:
```
#step 12 and 14 (Kurie plot)
print("At 0.7 A, the count is ~ 112, so we will regard this as the background")
y = []
```

```
x = []
for A in range(len(count)):
    y.append(np.sqrt((count[A])/newI[A]**3))
    x.append((np.sqrt((E_0**2) + (pc_e[A])**2) - E_0)/1000)
plt.plot(x,y)
plt.title("Kurie Plot")
plt.xlabel("E")
plt.ylabel("Intensity")
plt.show()
print("The max value of the energy is",max(x))
```

At 0.7 A, the count is ~ 112, so we will regard this as the background



The max value of the energy is 629.618403685

In [356]:
```
a = []
b = []
for N in range(len(x)):
    if x[N] >= 170 and x[N] <= 450:
        a.append(x[N])
        b.append(y[N])
print(linregress(a,b))
```

```
        print("This is the linear part of the Kurie plot, from x data points of 179 to 419.")
        print("The x-intercept would be",-8128/-15.189428097350406, "which is the E_max of the
```

```
LinregressResult(slope=-8.6313744584429077, intercept=4629.847732594646, rvalue=-0.9973577326551
This is the linear part of the Kurie plot, from x data points of 179 to 419.
The x-intercept would be 535.1090210840671 which is the E_max of the beta decay which is
lower than expected value due to possible errors with nonrealistic simulation
```

In [345]:
```
#step 13
display(Image(filename='tabulation.PNG'))
print("Step 14 was part of step 12, see above")
```

I	A	B	Brc	x	y
204	1571	0.0305178330	348574.6586	107.5665374	7434.599072
222.7	1635	0.0333153010	380527.3357	126.1194516	6649.565446
250.3	1953	0.0374441843	427687.4365	155.3606283	6099.223656
272	2128	0.0406904440	464766.2115	179.7440981	5620.140432
304.8	2127	0.0455972329	520811.5488	218.6329095	4736.701454
328.4	2165	0.0491277273	561136.8524	247.9426073	4273.059493
348.3	2129	0.0521047120	595139.9686	273.4174418	3879.469402
371.3	2023	0.0555454481	634440.0527	303.6367999	3435.779067
391.7	1918	0.0585972314	669297.5185	331.067644	3087.512012
434.5	1374	0.0649999924	742429.8489	390.2882039	2236.780701
454.8	1085	0.0680368160	777116.4448	419.0688787	1856.089427
494.2	630	0.0739309465	844439.1975	476.0133339	1248.62169
505.2	574	0.0755765159	863234.8899	492.1411309	1153.123028
509.6	602	0.0762347437	870753.1668	498.6180896	1165.651773
517	531	0.0773417631	883397.5417	509.543361	1071.337132
525.5	579	0.0786133395	897921.4858	522.1410572	1091.678892
532.2	580	0.0796156409	909369.7711	532.1063044	1072.053362
546	653	0.0816800825	932949.8216	552.7258157	1094.667787
556.6	1013	0.0832658131	951062.0342	568.6462968	1324.660638
565.2	1684	0.0845523492	965756.8483	581.6130765	1669.100647
574.2	2174	0.0858987242	981135.142	595.2292346	1852.037173
583.9	2475	0.0873498172	997709.5253	609.9554626	1927.057177
596.8	2274	0.0892796214	1019751.746	629.6184037	1787.585076

```
Step 14 was part of step 12, see above
```

In [348]:
```
#step 15 -- different data
```

```
I = [146, 181.2, 196.8, 214.5, 231.4, 256.5, 295, 317.9, 351.1, 381.3, 433, 452.4, 479
    552.2, 574.1, 589.4, 594.3, 605.8, 609.8, 616.5, 625.2, 630.1, 635.7, 640.9, 648.7
count = [381, 498, 506, 541, 610, 654, 662, 778, 737, 642, 445, 337, 241, 198, 165, 21
        790, 562, 519, 429, 287, 243, 167, 155, 148, 138]
pc_e = []
for N in range(len(I)):
    pc_e.append(newI[N]*0.0381*c)
newI = []
for y in I:
    newI.append(y*0.00014959722084772217)
y = []
x = []
for A in range(len(count)):
    y.append(np.sqrt((count[A])/newI[A]**3))
    x.append((np.sqrt((E_0**2) + (pc_e[A])**2) - E_0)/1000)
plt.plot(x,y)
plt.title("Kurie Plot")
plt.xlabel("E")
plt.ylabel("Intensity")
plt.show()
```