multiplicityFunction

February 16, 2022

0.1 import statements

[5]:
```
%matplotlib widget
```

[6]:
```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import scipy.special as sp
```

[7]:
```
pd.set_option('display.max_colwidth', None)
pd.set_option('display.max_columns', None)
pd.set_option('display.max_rows', None)
```

0.2 define some helpful functions

[8]:
```
def factorial(x):
    return sp.factorial(x, exact=True)

def multiplicityEinstein(N,q):
    return factorial(q+N-1)//factorial(q)//factorial(N-1)

def logArray(array):
    import math as math
    return [math.log(x) for x in array]
```

0.3 Two Einstein Solids

0.3.1 3 particles each and 6 energy units

[9]:
```
table1 = pd.DataFrame({'q_A':range(0, 6+1, 1),'q_B':range(6,0-1,-1)})
table1['multi_A'] = [multiplicityEinstein(3,i) for i in table1['q_A']]
table1['multi_B'] = [multiplicityEinstein(3,i) for i in table1['q_B']]
table1['multi_total'] = table1['multi_A']*table1['multi_B']

table1
```

[9]:

	q_A	q_B	multi_A	multi_B	multi_total
0	0	6	1	28	28
1	1	5	3	21	63
2	2	4	6	15	90
3	3	3	10	10	100
4	4	2	15	6	90
5	5	1	21	3	63
6	6	0	28	1	28

how many total microstates from the above situation? First just add up the microstate from each macrostate

[10]:
```
table1['multi_total'].sum()
```

[10]:
```
462
```

Alternatively, treat the two Einstein solids as one solid with the combined number of particles

[11]:
```
multiplicityEinstein(6,6)
```

[11]:
```
462
```

most likely macrostate just sort the table for the macrostate with the highest total multiplicity.

[12]:
```
table1.sort_values('multi_total', ascending=False).iloc[0]
```

[12]:
```
q_A              3
q_B              3
multi_A         10
multi_B         10
multi_total    100
Name: 3, dtype: int64
```

[14]:
```
table1['probability'] = table1['multi_total']/table1['multi_total'].sum()
```

[15]:
```
table1
```

[15]:

	q_A	q_B	multi_A	multi_B	multi_total	probability
0	0	6	1	28	28	0.060606

1	1	5	3	21	63	0.136364
2	2	4	6	15	90	0.194805
3	3	3	10	10	100	0.216450
4	4	2	15	6	90	0.194805
5	5	1	21	3	63	0.136364
6	6	0	28	1	28	0.060606

[17]:
```
fig1 = plt.figure()
ax1 = fig1.add_subplot()

ax1.bar(table1['q_A'],table1['multi_total'])
```

```
Canvas(toolbar=Toolbar(toolitems=[('Home', 'Reset original view', 'home',␣
↪'home'), ('Back', 'Back to previous …
```

[17]:
```
<BarContainer object of 7 artists>
```

0.4 Now prepare a function to scale this up to large numbers

[21]:
```
def multiTable(N_a, N_b, q):
    df = pd.DataFrame({'q_A':range(0, q+1, 1),'q_B':range(q, 0-1,-1)})
    df['multi_A'] = [multiplicityEinstein(N_a, i) for i in df['q_A']]
    df['multi_B'] = [multiplicityEinstein(N_b, i) for i in df['q_B']]
    df['multi_total'] = df['multi_A']*df['multi_B']
    df['probability'] = df['multi_total']/df['multi_total'].sum()
    return df
```

0.4.1 Try with Na = 300, Nb = 200, q = 100

[51]:
```
table2 = multiTable(300, 200, 100)
```

[52]:
```
fig0 = plt.figure()
ax0 = fig0.add_subplot(111)

ax0.plot(table2['q_A'], table2['probability'])
```

```
Canvas(toolbar=Toolbar(toolitems=[('Home', 'Reset original view', 'home',␣
↪'home'), ('Back', 'Back to previous …
```

[52]:
```
[<matplotlib.lines.Line2D at 0x7f171fb69310>]
```

Make these numbers floats instead of integers so we can read them better. Note that this will break for numbers much larger than this.

[53]:
```
table2 = table2.astype(float)
```

Total multiplicity of all macrostates:

[54]:
```
total = table2['multi_total'].sum()
total
```

[54]:
```
9.261760158884879e+115
```

Total mulitplicity of all states as a combined Einstein solid.

[55]:
```
float(multiplicityEinstein(500,100))
```

[55]:
```
9.261760158884879e+115
```

Most probable macrostate

[56]:
```
table2.sort_values('multi_total', ascending=False).iloc[0]
```

[56]:
```
q_A             6.000000e+01
q_B             4.000000e+01
multi_A         1.303375e+69
multi_B         5.268097e+45
multi_total    6.866305e+114
probability     7.413607e-02
Name: 60, dtype: float64
```

Multiplicity of the most common macrostate:

[57]:
```
multi_max = table2['multi_total'].sort_values(ascending=False).iloc[0]
multi_max
```

[57]:
```
6.866305444480905e+114
```

[58]:
```
table2[55:65]
```

[58]:

	q_A	q_B	multi_A	multi_B	multi_total	probability
55	55.0	45.0	1.473125e+65	2.982131e+49	4.393050e+114	0.047432
56	56.0	44.0	9.338557e+65	5.499832e+48	5.136049e+114	0.055454
57	57.0	43.0	5.832502e+66	9.958543e+47	5.808323e+114	0.062713
58	58.0	42.0	3.590006e+67	1.769493e+47	6.352491e+114	0.068588
59	59.0	41.0	2.178343e+68	3.083764e+46	6.717494e+114	0.072529
60	60.0	40.0	1.303375e+69	5.268097e+45	6.866305e+114	0.074136
61	61.0	39.0	7.692049e+69	8.816899e+44	6.782001e+114	0.073226
62	62.0	38.0	4.478757e+70	1.444786e+44	6.470846e+114	0.069866
63	63.0	37.0	2.573508e+71	2.316534e+43	5.961620e+114	0.064368
64	64.0	36.0	1.459662e+72	3.631855e+42	5.301279e+114	0.057238

[ ]:

[ ]:

0.4.2 Now to add in the Entropy

[59]:
```
table2['S_a'] = logArray(table2['multi_A'])
table2['S_b'] = logArray(table2['multi_B'])
table2['S_total'] = logArray(table2['multi_total'])
```

[60]:
```
table2
```

[60]:

	q_A	q_B	multi_A	multi_B	multi_total	probability	\
0	0.0	100.0	1.000000e+00	2.772168e+81	2.772168e+81	2.993133e-35	
1	1.0	99.0	3.000000e+02	9.271464e+80	2.781439e+83	3.003143e-33	
2	2.0	98.0	4.515000e+04	3.080117e+80	1.390673e+85	1.501521e-31	
3	3.0	97.0	4.545100e+06	1.016335e+80	4.619344e+86	4.987544e-30	
4	4.0	96.0	3.442913e+08	3.330557e+79	1.146682e+88	1.238082e-28	
5	5.0	95.0	2.093291e+10	1.083842e+79	2.268798e+89	2.449640e-27	
6	6.0	94.0	1.064090e+12	3.502212e+78	3.726667e+90	4.023714e-26	
7	7.0	93.0	4.651592e+13	1.123576e+78	5.226419e+91	5.643009e-25	
8	8.0	92.0	1.785049e+15	3.578514e+77	6.387821e+92	6.896984e-24	
9	9.0	91.0	6.108833e+16	1.131351e+77	6.911237e+93	7.462120e-23	
10	10.0	90.0	1.887629e+18	3.550103e+76	6.701278e+94	7.235426e-22	
11	11.0	89.0	5.319683e+19	1.105568e+76	5.881273e+95	6.350059e-21	
12	12.0	88.0	1.378684e+21	3.416513e+75	4.710294e+96	5.085744e-20	
13	13.0	87.0	3.308843e+22	1.047572e+75	3.466251e+97	3.742540e-19	
14	14.0	86.0	7.397627e+23	3.186670e+74	2.357380e+98	2.545283e-18	
15	15.0	85.0	1.548570e+25	9.615917e+73	1.489092e+99	1.607785e-17	
16	16.0	84.0	3.048747e+26	2.878003e+73	8.774304e+99	9.473689e-17	
17	17.0	83.0	5.667082e+27	8.542483e+72	4.841096e+100	5.226972e-16	
18	18.0	82.0	9.980362e+28	2.514277e+72	2.509339e+101	2.709355e-15	
19	19.0	81.0	1.670397e+30	7.337036e+71	1.225577e+102	1.323265e-14	
20	20.0	80.0	2.664284e+31	2.122500e+71	5.654942e+102	6.105688e-14	
21	21.0	79.0	4.059861e+32	6.086021e+70	2.470840e+103	2.667787e-13	
22	22.0	78.0	5.923706e+33	1.729481e+70	1.024494e+104	1.106154e-12	
23	23.0	77.0	8.293189e+34	4.870018e+69	4.038798e+104	4.360724e-12	
24	24.0	76.0	1.116125e+36	1.358664e+69	1.516439e+105	1.637312e-11	
25	25.0	75.0	1.446498e+37	3.754854e+68	5.431389e+105	5.864317e-11	
26	26.0	74.0	1.808123e+38	1.027789e+68	1.858368e+106	2.006495e-10	
27	27.0	73.0	2.183141e+39	2.785947e+67	6.082114e+106	6.566910e-10	
28	28.0	72.0	2.549596e+40	7.476991e+66	1.906331e+107	2.058281e-09	
29	29.0	71.0	2.883681e+41	1.986507e+66	5.728452e+107	6.185058e-09	
30	30.0	70.0	3.162437e+42	5.223777e+65	1.651987e+108	1.783664e-08	
31	31.0	69.0	3.366465e+43	1.359347e+65	4.576195e+108	4.940956e-08	
32	32.0	68.0	3.482188e+44	3.499812e+64	1.218700e+109	1.315841e-07	
33	33.0	67.0	3.503292e+45	8.913378e+63	3.122616e+109	3.371515e-07	
34	34.0	66.0	3.431165e+46	2.245099e+63	7.703305e+109	8.317323e-07	
35	35.0	65.0	3.274312e+47	5.591567e+62	1.830854e+110	1.976788e-06	
36	36.0	64.0	3.046929e+48	1.376712e+62	4.194743e+110	4.529099e-06	
37	37.0	63.0	2.766941e+49	3.350173e+61	9.269731e+110	1.000861e-05	

38	38.0	62.0	2.453840e+50	8.055759e+60	1.976754e+111	2.134318e-05
39	39.0	61.0	2.126661e+51	1.913629e+60	4.069640e+111	4.394024e-05
40	40.0	60.0	1.802345e+52	4.489667e+59	8.091931e+111	8.736925e-05
41	41.0	59.0	1.494628e+53	1.040077e+59	1.554529e+112	1.678437e-04
42	42.0	58.0	1.213495e+54	2.378471e+58	2.886264e+112	3.116324e-04
43	43.0	57.0	9.651522e+54	5.367756e+57	5.180702e+112	5.593647e-04
44	44.0	56.0	7.523800e+55	1.195165e+57	8.992179e+112	9.708931e-04
45	45.0	55.0	5.751527e+56	2.624675e+56	1.509589e+113	1.629916e-03
46	46.0	54.0	4.313645e+57	5.683351e+55	2.451596e+113	2.647009e-03
47	47.0	53.0	3.175577e+58	1.213047e+55	3.852126e+113	4.159172e-03
48	48.0	52.0	2.295678e+59	2.551250e+54	5.856849e+113	6.323689e-03
49	49.0	51.0	1.630400e+60	5.285459e+53	8.617411e+113	9.304290e-03
50	50.0	50.0	1.138019e+61	1.078234e+53	1.227050e+114	1.324857e-02
51	51.0	49.0	7.809934e+61	2.165128e+52	1.690951e+114	1.825733e-02
52	52.0	48.0	5.271706e+62	4.277873e+51	2.255169e+114	2.434925e-02
53	53.0	47.0	3.501208e+63	8.313276e+50	2.910651e+114	3.142654e-02
54	54.0	46.0	2.288753e+64	1.588309e+50	3.635246e+114	3.925006e-02
55	55.0	45.0	1.473125e+65	2.982131e+49	4.393050e+114	4.743213e-02
56	56.0	44.0	9.338557e+65	5.499832e+48	5.136049e+114	5.545435e-02
57	57.0	43.0	5.832502e+66	9.958543e+47	5.808323e+114	6.271295e-02
58	58.0	42.0	3.590006e+67	1.769493e+47	6.352491e+114	6.858837e-02
59	59.0	41.0	2.178343e+68	3.083764e+46	6.717494e+114	7.252935e-02
60	60.0	40.0	1.303375e+69	5.268097e+45	6.866305e+114	7.413607e-02
61	61.0	39.0	7.692049e+69	8.816899e+44	6.782001e+114	7.322584e-02
62	62.0	38.0	4.478757e+70	1.444786e+44	6.470846e+114	6.986626e-02
63	63.0	37.0	2.573508e+71	2.316534e+43	5.961620e+114	6.436811e-02
64	64.0	36.0	1.459662e+72	3.631855e+42	5.301279e+114	5.723836e-02
65	65.0	35.0	8.174106e+72	5.563692e+41	4.547821e+114	4.910320e-02
66	66.0	34.0	4.520528e+73	8.321762e+40	3.761876e+114	4.061729e-02
67	67.0	33.0	2.469423e+74	1.214334e+40	2.998705e+114	3.237727e-02
68	68.0	32.0	1.332762e+75	1.727286e+39	2.302061e+114	2.485555e-02
69	69.0	31.0	7.108064e+75	2.392777e+38	1.700801e+114	1.836369e-02
70	70.0	30.0	3.746965e+76	3.225047e+37	1.208414e+114	1.304735e-02
71	71.0	29.0	1.952644e+77	4.224953e+36	8.249828e+113	8.907409e-03
72	72.0	28.0	1.006154e+78	5.373844e+35	5.406914e+113	5.837890e-03
73	73.0	27.0	5.127250e+78	6.628530e+34	3.398613e+113	3.669511e-03
74	74.0	26.0	2.584411e+79	7.919040e+33	2.046606e+113	2.209737e-03
75	75.0	25.0	1.288760e+80	9.150890e+32	1.179330e+113	1.273332e-03
76	76.0	24.0	6.359012e+80	1.021305e+32	6.494489e+112	7.012154e-04
77	77.0	23.0	3.105180e+81	1.099162e+31	3.413096e+112	3.685148e-04
78	78.0	22.0	1.500837e+82	1.138771e+30	1.709110e+112	1.845341e-04
79	79.0	21.0	7.181220e+82	1.133619e+29	8.140765e+111	8.789652e-05
80	80.0	20.0	3.402103e+83	1.082091e+28	3.681383e+111	3.974821e-05
81	81.0	19.0	1.596048e+84	9.882105e+26	1.577232e+111	1.702950e-05
82	82.0	18.0	7.415785e+84	8.612844e+25	6.387100e+110	6.896206e-06
83	83.0	17.0	3.413048e+85	7.144295e+24	2.438382e+110	2.632742e-06
84	84.0	16.0	1.556187e+86	5.622825e+23	8.750169e+109	9.447631e-07

85	85.0	15.0	7.030306e+86	4.184428e+22	2.941781e+109	3.176265e-07
86	86.0	14.0	3.147288e+87	2.933010e+21	9.231027e+108	9.966817e-08
87	87.0	13.0	1.396383e+88	1.927800e+20	2.691947e+108	2.906518e-08
88	88.0	12.0	6.140911e+88	1.182141e+19	7.259426e+107	7.838063e-09
89	89.0	11.0	2.677161e+89	6.723080e+17	1.799877e+107	1.943342e-09
90	90.0	10.0	1.157129e+90	3.521613e+16	4.074959e+106	4.399768e-10
91	91.0	9.0	4.959123e+90	1.684982e+15	8.356034e+105	9.022080e-11
92	92.0	8.0	2.107627e+91	7.290789e+13	1.536627e+105	1.659109e-11
93	93.0	7.0	8.883762e+91	2.817696e+12	2.503174e+104	2.702698e-12
94	94.0	6.0	3.714169e+92	9.574696e+10	3.556203e+103	3.839663e-13
95	95.0	5.0	1.540403e+93	2.802350e+09	4.316747e+102	4.660828e-14
96	96.0	4.0	6.338115e+93	6.868505e+07	4.353337e+101	4.700335e-15
97	97.0	3.0	2.587519e+94	1.353400e+06	3.501948e+100	3.781083e-16
98	98.0	2.0	1.048209e+95	2.010000e+04	2.106901e+99	2.274838e-17
99	99.0	1.0	4.214013e+95	2.000000e+02	8.428026e+97	9.099810e-19
100	100.0	0.0	1.681391e+96	1.000000e+00	1.681391e+96	1.815412e-20

	S_a	S_b	S_total
0	0.000000	187.529022	187.529022
1	5.703782	186.433749	192.137531
2	10.717746	185.331775	196.049521
3	15.329560	184.223010	199.552571
4	19.656999	183.107362	202.764361
5	23.764589	181.984735	205.749323
6	27.693141	180.855032	208.548173
7	31.470816	179.718154	211.188970
8	35.118222	178.574000	213.692222
9	38.651097	177.422465	216.073562
10	42.081853	176.263444	218.345297
11	45.420530	175.096827	220.517357
12	48.675417	173.922503	222.597919
13	51.853470	172.740357	224.593828
14	54.960616	171.550273	226.510890
15	58.001959	170.352132	228.354091
16	60.981943	169.145809	230.127752
17	63.904472	167.931178	231.835650
18	66.773002	166.708112	233.481114
19	69.590614	165.476477	235.067091
20	72.360073	164.236136	236.596209
21	75.083872	162.986951	238.070823
22	77.764270	161.728778	239.493048
23	80.403328	160.461469	240.864797
24	83.002926	159.184874	242.187800
25	85.564794	157.898836	243.463630
26	88.090523	156.603196	244.693718
27	90.581583	155.297789	245.879372
28	93.039339	153.982447	247.021785

29	95.465057	152.656994	248.122050
30	97.859917	151.321252	249.181169
31	100.225022	149.975036	250.200058
32	102.561405	148.618155	251.179560
33	104.870032	147.250414	252.120446
34	107.151814	145.871610	253.023425
35	109.407607	144.481535	253.889143
36	111.638219	143.079974	254.718192
37	113.844412	141.666703	255.511115
38	116.026909	140.241493	256.268402
39	118.186393	138.804107	256.990500
40	120.323514	137.354299	257.677813
41	122.438887	135.891816	258.330703
42	124.533100	134.416393	258.949493
43	126.606711	132.927760	259.534471
44	128.660251	131.425634	260.085886
45	130.694231	129.909722	260.603953
46	132.709134	128.379721	261.088855
47	134.705425	126.835316	261.540741
48	136.683549	125.276179	261.959727
49	138.643931	123.701969	262.345900
50	140.586980	122.112334	262.699314
51	142.513087	120.506904	263.019991
52	144.422630	118.885296	263.307925
53	146.315969	117.247108	263.563077
54	148.193453	115.591924	263.785377
55	150.055417	113.919308	263.974724
56	151.902183	112.228802	264.130985
57	153.734062	110.519930	264.253992
58	155.551355	108.792193	264.343548
59	157.354351	107.045065	264.399416
60	159.143328	105.277998	264.421327
61	160.918559	103.490414	264.408973
62	162.680302	101.681705	264.362007
63	164.428812	99.851231	264.280043
64	166.164331	97.998317	264.162649
65	167.887098	96.122251	264.009349
66	169.597341	94.222278	263.819618
67	171.295281	92.297600	263.592881
68	172.981135	90.347370	263.328505
69	174.655112	88.370688	263.025800
70	176.317413	86.366596	262.684009
71	177.968236	84.334071	262.302308
72	179.607772	82.272022	261.879794
73	181.236207	80.179276	261.415483
74	182.853720	78.054578	260.908298
75	184.460488	75.896574	260.357062

76	186.056680	73.703804	259.760484
77	187.642464	71.474686	259.117150
78	189.218001	69.207503	258.425503
79	190.783447	66.900383	257.683829
80	192.338956	64.551277	256.890234
81	193.884679	62.157938	256.042617
82	195.420759	59.717882	255.138641
83	196.947339	57.228356	254.175695
84	198.464557	54.686291	253.150848
85	199.972548	52.088242	252.060790
86	201.471444	49.430316	250.901760
87	202.961373	46.708081	249.669455
88	204.442461	43.916444	248.358906
89	205.914830	41.049493	246.964323
90	207.378600	38.100281	245.478881
91	208.833887	35.060531	243.894419
92	210.280806	31.920218	242.201024
93	211.719469	28.666941	240.386409
94	213.149983	25.284975	238.434958
95	214.572457	21.753724	236.326182
96	215.986995	18.045042	234.032037
97	217.393698	14.118131	231.511829
98	218.792667	9.908475	228.701142
99	220.183999	5.298317	225.482317
100	221.567790	0.000000	221.567790

Plotting Entropy

[61]:
```
fig1 = plt.figure()
ax1 = fig1.add_subplot(111)

ax1.plot(table2['q_A'], table2['S_a'], label='S_a')
ax1.plot(table2['q_A'], table2['S_b'], label='S_b')
ax1.plot(table2['q_A'], table2['S_total'], label='S_total')

# labels
ax1.set_ylabel('Entropy')
ax1.set_xlabel('q_A')

#legend
ax1.legend()
```

```
Canvas(toolbar=Toolbar(toolitems=[('Home', 'Reset original view', 'home',␣
↪'home'), ('Back', 'Back to previous …
```

[61]:
```
<matplotlib.legend.Legend at 0x7f171fa9bad0>
```

[ ]: