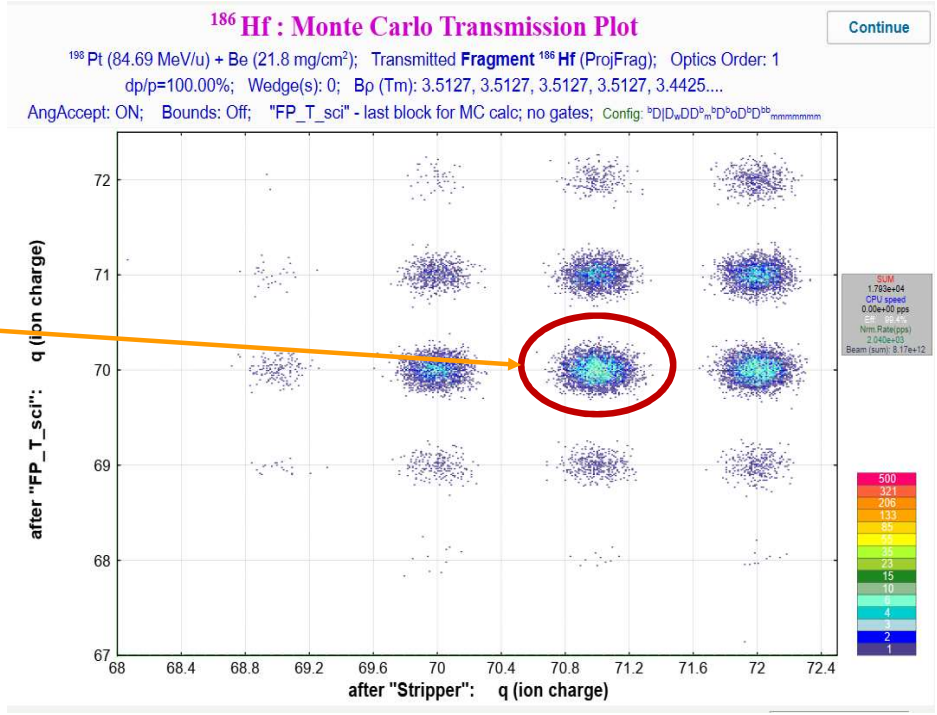
Monte Carlo Reader

Kenny Haak

This material is based upon work supported by the U.S. Department of Energy, Office of Science, Office of Nuclear Physics and used resources of the Facility for Rare Isotope Beams (FRIB) Operations, which is a DOE Office of Science User Facility under Award Number DE-SC0023633, and by the US National Science Foundation under Grants No. PHY-20-12040 and 23-10078 “Windows on the Universe: Open Quantum Systems in Atomic Nuclei at FRIB”.

Foreword

- This code was made to read the LISE spectra that are generated in the Monte Carlo transmission plots
- It benefitted my analysis to count blobs on the diagonal q_inital = q_final (x=y) line of the MC plot
- So we are really just trying to accomplish the function of answer the question:
  - How many counts are in this blob?
- Yet I aimed to automate it, because I had to sum the counts from over hundreds (and with mistakes, over thousands) of blobs
- Therefore these python scripts try to cut the spectra up into a grid and take the sum of the bins in each grid
- It beats having to draw contours



Initialize

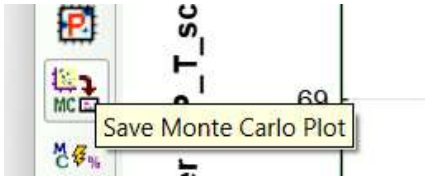
- Make sure you have the right code versions

```
import re
import numpy as np
import pandas as pd
import sys

print(np.__version__)
print(pd.__version__)
print(sys.version)
```

```
1.21.6
1.3.5
3.9.10 (tags/v3.9.10:f2f3f53, Jan 17 2022,
```

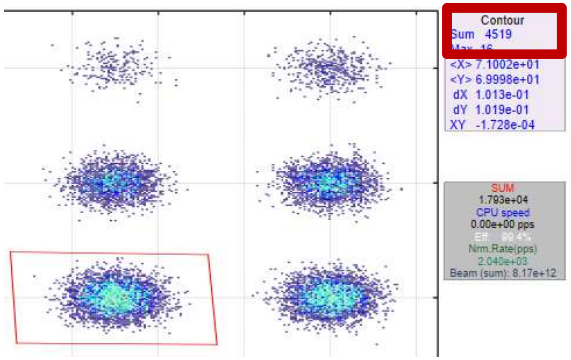
Counting 2D Monte Carlo

- Run your Monte Carlo in LISE (see my thesis for details)
- Click the save Monte Carlo plot button



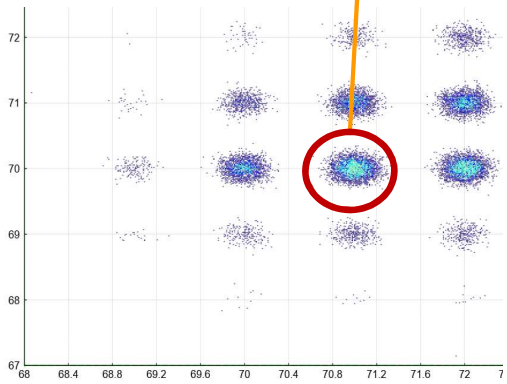
- Then simply run the ### 2D ### cell

```
################################## 2D
file = 'plotMC.spa'

with open(file, "r") as f:
    lines = f.readlines()

with open(file, "r") as a:
    soup = a.read()
```



Out[4]:

	68.0	69.0	70.0	71.0	72.0
72.0	0.0	2.0	48.0	210.0	484.0
71.0	1.0	28.0	681.0	2028.0	2528.0
70.0	0.0	149.0	2321.0	4519.0	4004.0
69.0	0.0	20.0	218.0	370.0	281.0
68.0	0.0	0.0	10.0	10.0	10.0
67.0	0.0	0.0	0.0	0.0	1.0



Final data structure is easily workable DataFrame

Use as necessary

Issues

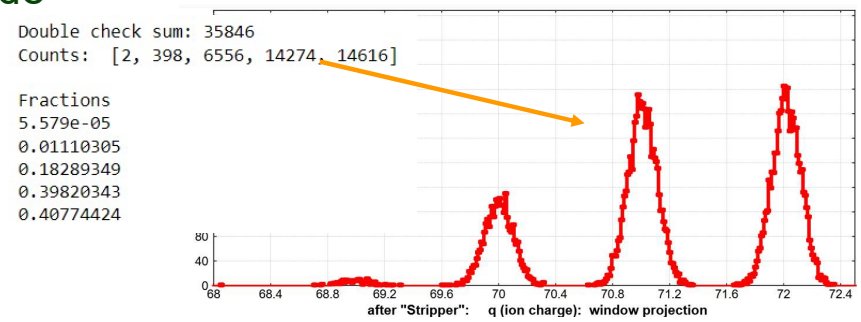
- When making actual read world data processing scripts there are many things that can go wrong
- One such issue here is when the halfway point between integers falls exactly on a bin
  - then you will get two indices for that halfway point
  - and all the others will only get one
  - and then the dimension is off
  - and you can't count boxes correctly
  - yada yada
- That is why there is a debug output that shows the construction of these indices (the step is being doubled and then the duplicates are being removed)

```
# For the issue 1: Half points being exaclty between steps
arr=np.array(yhIND)
_, unique_indices = np.unique(arr[:, 1], return_index=True)
yhIND = list(arr[sorted(unique_indices)].T[0].astype(np.int32))
print(yhIND)
```

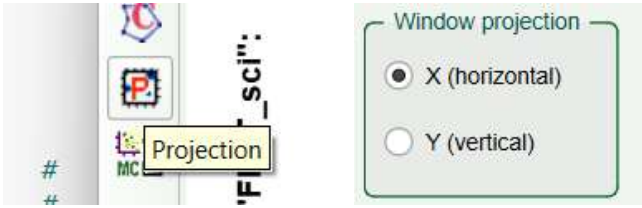
```
Yaxis Grid
67.4888888888889 67.5 0.011111111111105743 40
67.50111111111111 67.5 0.0011111111111148375 41
68.49111111111111 68.5 0.008888888888890278 122
68.50333333333333 68.5 0.0033333333333303017 123
69.49333333333334 69.5 0.006666666666660603 204
69.50555555555556 69.5 0.005555555555559977 205
70.49555555555555 70.5 0.004444444444445139 286
70.50777777777778 70.5 0.007777777777775441 287
71.49777777777778 71.5 0.002222222222215464 368
71.51 71.5 0.010000000000005116 369
[40, 122, 204, 286, 368]
Xaxis Grid
68.49647887323944 68.5 0.003521126760560378 47
68.50704225352112 68.5 0.007042253521120756 48
69.5 69.5 0.0 142
70.49295774647888 70.5 0.007042253521120756 236
70.50352112676056 70.5 0.003521126760560378 237
71.49647887323944 71.5 0.003521126760560378 331
71.50704225352112 71.5 0.007042253521120756 332
[47, 142, 236, 331]
```

Counting 1D Monte Carlo

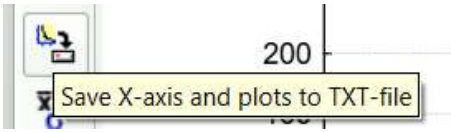
- The Monte Carlo in LISE plots in 2D
- But if you have three materials and you are trying to calculate charge state distributions with Monte Carlo you will have to count at least one of them in 1D
- In fact, counting each distribution individually and convoluting them is necessary
- This is a tedious process but if it is every at all necessary, you can at least count 1D peaks with this code



1. First go to projection of your 2D and select X or Y parameter



2. Next save the projection to text file



3. Run the ### 1D ### cell

```
################################## 1D ###
file = 'plot.txt'
def read1d(file):
```