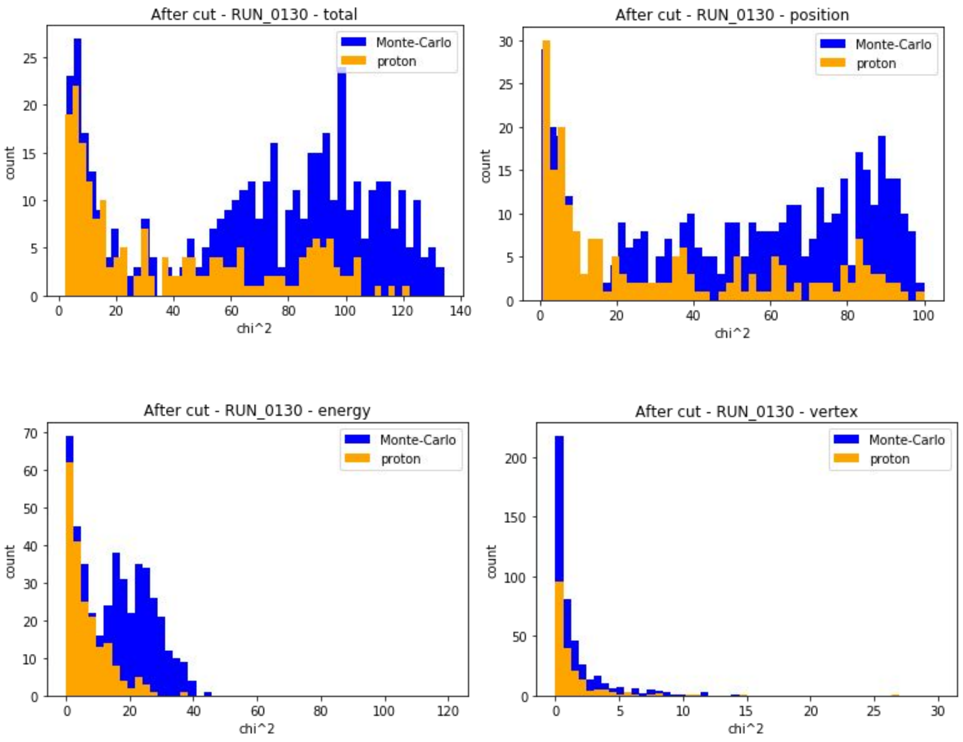
Weekly Report 9/18/2018

Christina Chen

Tasks

1. Finish fitting each component of chi^2 value for
    a. Monte Carlo
    b. Differential evolution
    c. Basinhopping
2. Tune parameters of differential evolution

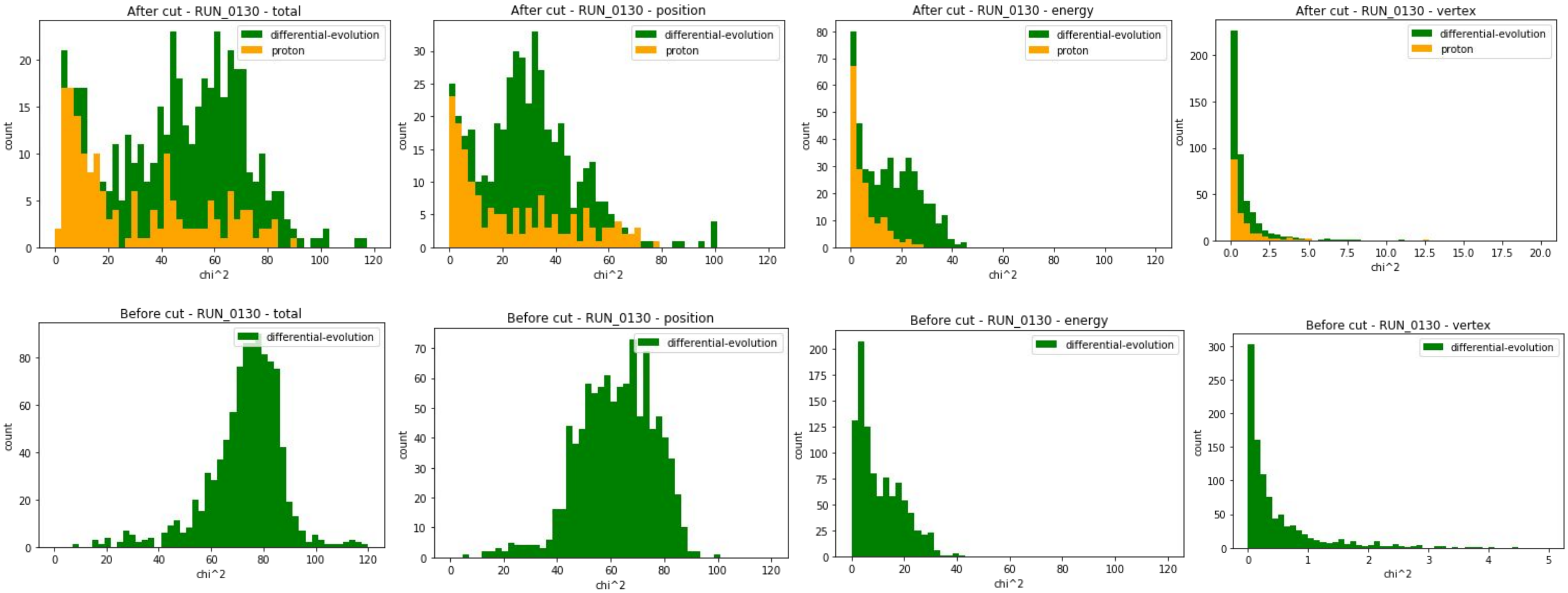
Continuing from last time: Monte Carlo (3 components)



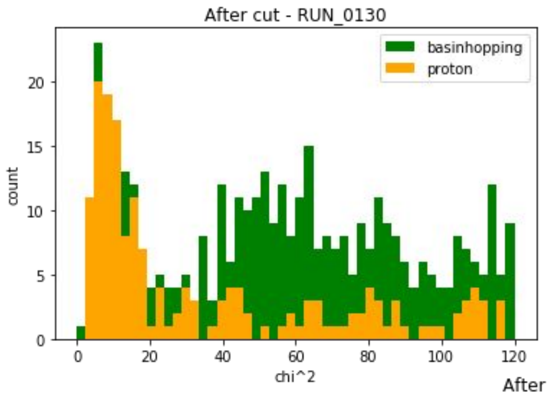
How to modify differential evolution (DE) and basin hopping (BH) code so that we could plot each individual component of the objective function?

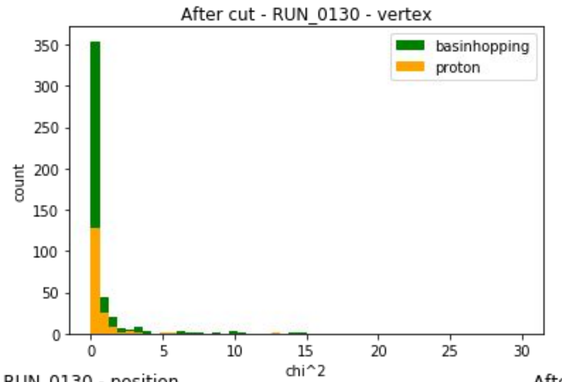
1. Instead of creating individual .h5 files for output of each chi^2 component, I used different group names within a single .h5 file to simplify the process
2. Only call “add_chi2 equals true” parameter in the objective function when the lowest chi^2 values have been reached

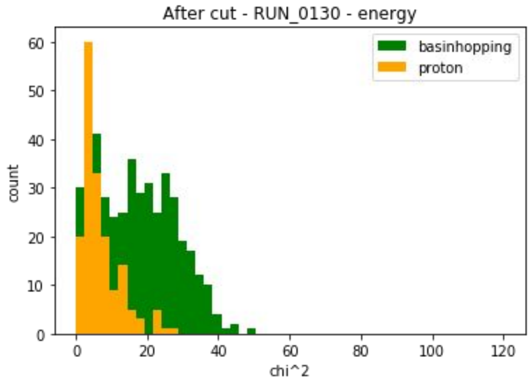
Differential Evolution

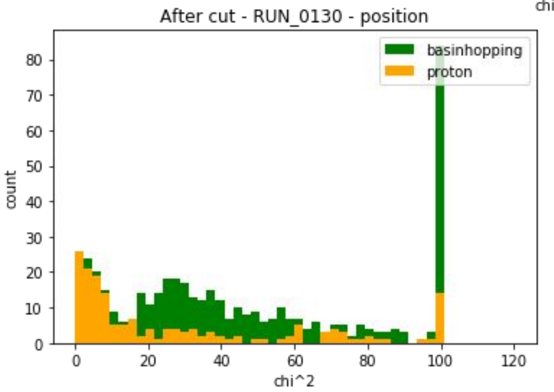


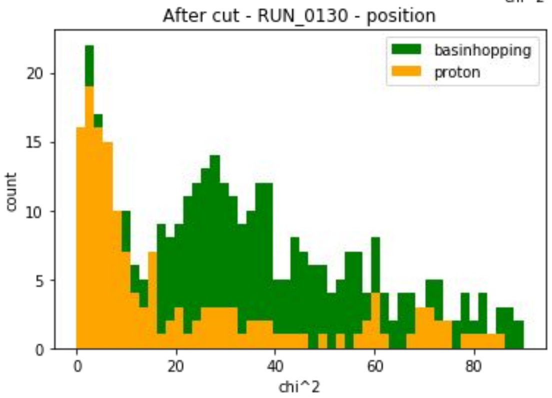
Basinhopping



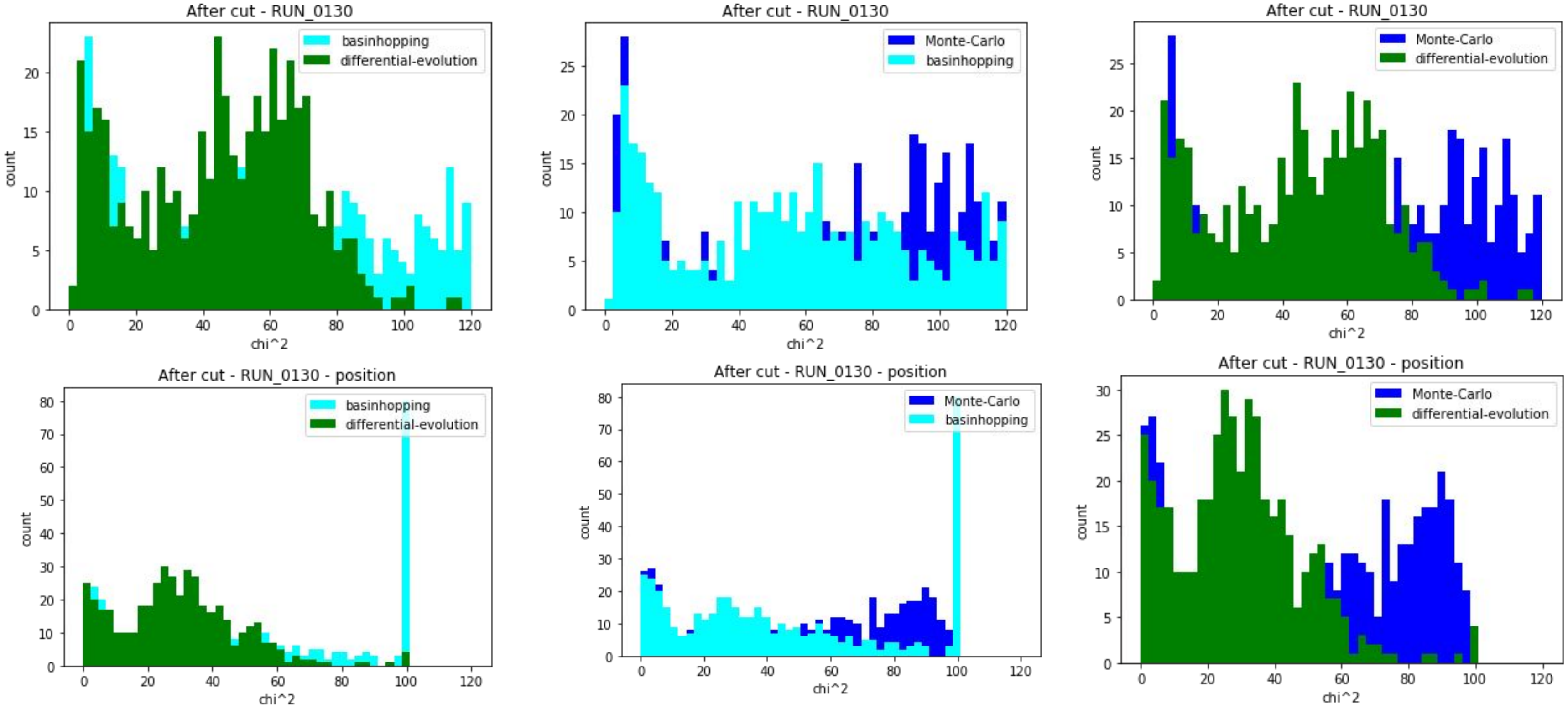








The 3 methods – total and position chi^2



Differential evolution – a closer look

scipy.optimize.differential_evolution(f, bounds, maxiter=1000, strategy='rand1bin',recombination=0.9, popsize=10, mutation=(0.5,1.0))

1. Initialization
    a. Define upper and lower **bounds** for each parameter, then randomly select initial parameter values. Size of **population** must >= 4.
2. Mutation
    a. Choose one point from population **x**, also known as the target vector (rand, best, etc.)
    b. Add the weighted difference between two randomly selected population members added to a third population member
    c. **V** = x1+**F**(x2-x3) for ‘1’. **V** is called the donor vector
3. Recombination
    a. Rand ~ [0,1], generated by binomial, exponential, etc.
    b. **T**i = **V**i if rand <= **recombination**; = xi if rand > **recombination**; **T** is the trial vector (this “crossover” is for each parameter individually)
4. Selection
    a. Compare **T** with **x** and select the one with lower chi^2 value to be a member of the next generation

Differential evolution in literature

Brownlee, Jason. "Differential Evolution." Clever Algorithms: Nature-Inspired Programming Recipes. Accessed September 18, 2018.
R. Storn and K. Price, "Differential evolution: A simple and efficient heuristic for global optimization over continuous spaces", Journal of Global Optimization, 1997.
Y. Ho and D. Pepyne. “Simple Explanation of the No Free Lunch Theorem of Optimization”. 2001.

1. Population size ~ 10*number of parameters
2. However NP above 40 does not substantially improve the convergence
3. Most popular strategies: “rand1bin” by Storn and Price (1997), “best2bin” improves population diversity for high NP
4. “Binomial is never worse than exponential” - Kenneth Price
5. F controls the amplification of differential variation; a value of 0.8 is suggested
6. F from (0.5,1.0) randomly from each generation significantly improves convergence behavior
7. Recombination (CR) suggested is 0.9; lower CR is preferable for independent parameters
8. If convergence is hard to be reached, make changes to the objective function
9. Finally, the “No Free Lunch Theorem of Optimization” tells us that “a general-purpose universal optimization strategy is impossible”

Tuning Parameters: 27/38 proton events (cleaned)

Strategy first: NP=15, RC=0.9, F=(0.5,1.0)

‘Best1bin’: 12.75s, chi^2 = 37.28

‘Rand1bin’: 100.3s, chi^2 = 25.40

‘Best2bin’: 109.87s, chi^2 = 48.9

Monte Carlo: chi^2 = 36.79

Population Size

RC=0.9, F=(0.5,1.0), strategy=”rand1bin”

NP=5: 13.27s, chi^2=37.03

NP=7: 34.68s, chi^2=37.8

NP=10: 44.64s, chi^2=26.97

NP=15: 100.3s, chi^2=25.40

NP=20: 153.58s, chi^2=27.79

Monte Carlo: chi^2 = 36.79

Mutation factor F ~ [0,2]

RC=0.9, strategy=”rand1bin”, NP=10

F=0.8: 63.99s, chi^2 = 24.00

F=0.9: ~67s, chi^2 = 22.04

F=1.0: ~46s, chi^2 = 39.84

Monte Carlo: chi^2 = 36.79