Code Reference of Runze's Project

Runze Zhao
with Prof. Huan Huang, Dr. Gang Wang and Ph.D's
University of California, Los Angeles
May 23, 2016

Abstract

My project includes two major parts: 1. search for possible existance of penta-quark particle $\Theta^{++}(uuud\bar{s})$, 2. study correlation functions (CF) of proton-kaon (pK) pairs. The used data comes from $\sqrt{S_{NN}} = 200$ GeV Au+Au collisions in STAR detector at RHIC. The decay channels of the penta-quark that I checked are $\Theta{+}{+} \to K^+ + p$ and its anti-particle part. The correlation study also uses those two channels. In order to be complete, opposite-charge channels ($K^- - p$ & $K^+ - \bar{p}$) are built as well. The signal of formation of $\Lambda(1520)$ is expected to see in both invariant mass distribution and correlation functions of opposite-sign pairs. Both Rotational Background (**RB**) and Mixed-Event Background (**MB**) are used in mass reconstruction. Only MB is used in correlation function in order to avoid any effect in same event. The methods to build those backgrounds are explained in detailed along with correponding code lines in Section **MS_PCF.C**.

PHYSICS AND DATA

The information of particles produced are saved in a tree structure in root files. Each entry in the tree represents an event, attached with leaves that records both event information (stored once for each entry) like Run ID, Referecne Multiplicity, magnetic field etc., and particle information (stored for multiple times for each entry) like pt, phi, Eta etc. Event information for particles in the same event is the same. Particle information, in principle, is unique for each particle. In each entry (event), the leaves of particle information is stored (in substructure of the leaf) n times for n particles in this event.

To select protons and kaons that we need in analysis, we use following cuts to recognize the identity of particles: (same cuts apply to their anti-particles up to opposite electric charge)

Proton:
DCA $\leq$ 1 cm
0.2 $\leq$ Pt $\leq$ 2.8 GeV/c
|Eta| $\leq$ 0.5
0 $<$ Flag $\leq$ 1000
TOFflag $\geq$ 1 && |TofYLocal| $\leq$ 1.8 && 0.8 $\leq TofM^2 \leq$ 1
ndEdx $\geq$ 10
$|n\sigma_p| \leq 2$

Kaon:
TOFflag $\geq$ 1
|Eta| $\leq$ 0.5
DCA $\leq$ 1 cm
1 $<$ Flag $\leq$ 1000

P $\leq$ 1.6 GeV/c && |TofYLocal| $\leq$ 1.8 && 0.2 $\leq Tof M^2 \leq$ 0.35

0.1 I. Penta-quark Θ^{++}

To search for Θ^{++}, whose mass is expected around 1.5 GeV, we reconstruct the invariant mass from $K^+ - pandK^- - \bar{p}$ pairs. The leaves provide momentum of particles so that we can build four vectors with energy $E = \sqrt{p^2 + m^2}$ with standard values of particle mass. By adding two four vectors we calculate the mass of Θ^{++} through the same equation. Larger centrality represents the larger overlap area of colliding Au ions.

0.2 II. Correlation function

Correlation function of a certain physical quantity is defined as ratio of foreground entry over background entry:

$$CF_Q = \frac{FG_Q}{BG_Q} \tag{1}$$

where Q is the quantity we want to look at, FG is the entry of foreground and BG is that of background.

We studied correlation functions of two quantities: 1. Q_{inv} 2. k^*. Q_{inv} is the module of difference of the two four vectors: $\sqrt{\Delta E^2 - \Delta P^2}$. k^* is the magnitude of momentum of either particle in central -mass frame. Sicne nothing interesting is found in Q_{inv} CF, the CF code will include only k^* part. The code for Q_{inv} would be the same except a one-line code of calculation / function.

To give an example of calculation of expectation on the effect from the formation of $\Lambda(1520)$, we will calculate here the expected value of k^* of opposite-sign pairs. In the central mass frame:

$$\begin{pmatrix} E_K \\ P_K \end{pmatrix} + \begin{pmatrix} E_p \\ P_p \end{pmatrix} \rightarrow \begin{pmatrix} E_\Lambda \\ 0 \end{pmatrix} \tag{2}$$

where $k^* = P_p = -P_K > 0$. Now we have $m_\Lambda = E_K + E_p = \sqrt{k^{*2} + m_K^2} + \sqrt{k^{*2} + m_p^2}$. After simple algebra, we get expected k^*:

$$\begin{aligned} k^* &= \sqrt{\frac{(m_\Lambda^2 + m_K^2 - m_p^2)^2}{4m_\Lambda^2} - m_K^2} \\ &= \frac{\sqrt{m_\Lambda^4 + m_K^4 + m_p^4 - 2m_\Lambda m_K - 2m_\Lambda m_p - 2m_K m_p}}{2m_\Lambda} \end{aligned}$$

Applying the standard values of masses of the three particles, we get $k^* \approx 0.243$ GeV/c.

****

Note: Below are the codes produced in this project. The way to run a code named **macro_name.C** is:
"$: nohup root -b -q macro_name.C++ > &".
Sometimes you need to use ".C" instead of "C++".
This is for the long-term macros that need data sets as input. It can run in the back and will create a log file named "nohup.out" recording any ouput information by root.
For short-term macros that only deals with simple histograms and root files in a short time, you can directly use "$: root -b macro_name.C".

The order of applying those codes is: MS_PCF.C → MS_both.C & PCF_both.C → Compare.C → PCF_Pt.C → Lednicky's code (main_in_C.cpp) → PCF_Compare.C

****

1. MS_PCF.C

Directory: "/home/runze/Analysis/KP_MS_PCF/MS_PCF.C"
This is the major code which produces the invariant mass distribution and momentum correlation fucntion.

line 1-22:
Add some classes, most are in root. To add the last two, remember to put folder src/ and file MyRef.h to the right directory and then change the address here.

line 28-43:
Some cut values and standard values.
xmin xmax: bottom and up bounds of invariant mass (called as MS below) histogram range. unit: GeV (set c=1 here and below)
pmin pmax: bottom and up bounds of momentum correlation function (PCF) histogram range. unit: GeV
Bin_PerUnit: number of bins in range of 1GeV you would like to have for MS and PCF histograms.
mk mp: mass of kaon and proton (GeV).
PI = π, **phi** = $\frac{\pi}{3}$.
Vz_cut and line42: range of Vz and Reference Multiplicity (RefM) used by buffer in method of MB.
Vz is the z-axis coordinate of decay vertex (the position decay happens).
RefM is the reference value of number of particles produced in an event/collision. It's used to define Centrality (cen).
Note: Need to change range and size (of each buffer grid) of RefM whenever cen is changed. Make sure not to set RefMsize too small so that buffer memory is too large. I checked the corresponding range of RefM for each cen: (consistant with line 43 provided by predecessors) cen = 1:[9,21]; 2: [21,42]; 3: [41,75]; 4: [72,123]; 5: [119,191]; 6: [183,278]; 7: [267,392]; 8:

[377,461]; 9: [444,622].
other constants: some of the cut values to identify proton and kaons.

line 45-58:
Run ID's of bad runs provided by predecessors.
line 61:
Function to calculate k^* from lab frame. Inputs are four vectors of the two particles. The definition is at line 544-563.
line 544-563:
Par: decay parent. **Boo**: boost four vector used for Lorentz Transformation (LT).
boost(), px(), etc.: functions from header "StLorentzVectorD.h", used to do LT, find momentum on x-axis, etc.

line 64 & 89:
main function. Inputs are
MB: index of mean-bias (index of data set, there are three: MB1,5,6)
list: index of one of ten lists of a data set.
cen: Centrality want to study in this run.
InputFileList: address of the .list file that store addresses of data (root files)
Note: The first two inputs should be changed whenever the 4th is changed. You can change this part (first two inputs and line 89) as you like because they are only used in line 89 for ouput file name. e.g. put all data file addresses into a total list and then you dont need the first two indices.

line 73-85:
Add TTree sturctures from root files in **InputFileList** into a chain. We loop the chain below when we want to loop through all events.
line 87-90: create an output file.
line 91-159: declare some variables and histograms.
"KPPM" represents Kaon, positive charge (Plus), Proton, negative charge (Minus) respectively, and similar for the other three.

line 161-193:
Creat a buffer to store particles (protons in this case) from other events. This is used in MB method. Divide the buffer into various grids according to Vz and RefM (variable name is Bz here because prevously use Bz instead of RefM). Each grid stores at most 10 other events. For each event we select no larger than 10 protons, . The buffer records, in each grid 1. number of events actually stored; 2. number of particles actually stored for each event; 3. magnetic field Bz (not used in this case); 4. particle charge (proton or anti-proton); 5. particle momentum. **Note** that buffers (array) of different kind of information have different dimensions. Buffer is initialized (to 0) right after being created.

line 195: get the number of total entries, i.e. the number of all events in input data set.
line 197: start loop all those events.
line 200: when each 10% data is looped, print (cout) a note. 10% comes from the 10 in

"nentries/10". Change it if needed.
line 201: read the (i+1)-th entry in the chain
line 203-228: get the leaves from the tree in chain and assign values to corresponding variables.

line 234-243: pass the data from bad runs.
line 245-268 and 287: define centrality and then cut to the centrality wanted.
line 271-288: fill some histograms and make some cuts to ensure the quality of data. 273 274 remove pile-up; 275 276 remove low quality tracks with Z-vertex.

line 290-303: read leaves recording particle information

line 305-310: set the curent event information in the buffer.
line 312-314: start the loop of MB: loop through all particles stored in the current entry (event)
line 315-326: assign values of particle information to variables.
line 328-338: make cuts so that the particles which survive from cutting are **Kaons**.
line 340: create the four vector **FourP02** of the Kaon from the informaiton.

line 342-345: start loop through all satisfied events, i.e. events with the same Vz and RefM as those of current event.
line 346-352:
for each satisfied event, loop through all stored particles (protons in this buffer) in the event; read particle information and create the four vector for **FourP01** the proton.

line 354-361:
get information like mass, $|k^*|$ (PCF) and rapidity of the decay parent, whose four vector is just the sum of the two of decay daughters.
line 362: make a cut on rapidity to ensure data of better quality.

line 364-381: fill the 8 MB histograms (four charge combinations, mass and PCF each) according to charges of the two particles. MB loop ends.

line 383-384:
start loop of foreground (**FG**) and RB. The first (outer most) loop is to select out all protons in current event.
line 385-298: get values of current particle information
line 400-415: fill some histograms and make cuts so that the survived particles are protons.

line 417: create a four vector **FourP** for the current proton in the loop.

line 420: start the 2nd loop to select out all kaons in current event.
line 421-432: get particle information of each particle in current event.
line 434-444: make cuts in similar way as in proton (1st) loop, so that the passed particles are Kaons.
line 446: create a four vector **FourP2** for current Kaon in the loop.

line 447-450: get mass, PCF (to be filled in FG histograms) and other particle information of the decay parent.

line 452-453:
start the loop to fill FG and RB. In this case, we rotate Kaon for 5 times, each time by phi = 60°; then collide each rotated Kaon with the original Proton in current event to get mass and PCF of produced decay parent, which are to be filled in RB histograms. Do the same steps to rotated Proton – current Kaon pairs and then fill into the same RB histograms.

line 455-456:
create four vector **FourP3** for Kaon rotated by kk*60°(integer kk $\in$ [1, 5]), and **FourP5** for rotated Proton.
line 458-467: collide FourP with FourP3, and FourP2 with FourP5 to get particle information of decay parent in RB.

line 469-514: fill the 8 FG and 8 BG histograms according to charge combinations.
Note that 1. the rapidity cut is included; 2. to avoid repitition that may enlarge statistics, only fill FG histograms when kk=1; 3. each RB histogram is filled twice for two combinations described above.

line 517-525:
replace the particle information in one event of the buffer zone with the same Bz and RefM of current event. All protons (or at most 10) in the current event are stored in the updated zone.

line 528-535: This is after loop of one event (entry) in the chain.
If the current event do have protons and is stored into the buffer, update the information in corresponding buffer zone (same Vz and RefM), e.g. number of protons stored in the buffer of current event, number of total events with same Vz and RefM ever stored in the buffer, etc.

line 538-542:
Indicate the current program is finished normally. Write all created histograms into the output root file. Close the output file and end the macro.

2. MS_BOTH.C

Directory:"/home/runze/Analysis/KP_MS_PCF/MS_both.C"
This is the code to normalize both mass background histograms (BG = RB / MB), then draw each of them with FG histogram together, and reduce FG by BG to search for signals (although nothing found :p).

line 14: the range for integral in normalization.
set the range outside the range of interested signal to avoid the contribution of signal itself.

line 15 and 106: This is the variable to decide how many bins you want to merge for a histogram.
If the histogram is over divided in **MS_PCF.C**, turn on the command at line 106 (currently commented out) and set value of this variable.

line 16 and 102-111: a function to draw FG and BG histograms on one canvas.
line 104: make clones of input foreground (FG) and background (BG) so that any operation on those histograms won't affect the original ones.
line 105: set FG line in red and BG line in blue.
line 106: Rebin both histograms (currently turned off).
line 107: use another function Scale() to normalize backgrounds.
line 108-110: Draw both histograms on the same canvas and return the canvas.

line 17 and 113-120: a function to normalize entries of histogram bg to the entries of fg in the range of x-axis [first, last].
line 115: find the bin numbers (global index) of head and end of the range in both histograms.
line 116: calculate the entries in the range for both histograms (fg and bg).
line 117-119: get the ratio for bg to be normalized with, and return the ratio (fg entry / bg entry).

line 19: inputs of the main macro, MB and cen, are for file naming only.
line 23-25: create names for input and output files. Read the input file.
line 27-41: read in all needed histograms (FG, RB, MB, each with 4 charge-pairs).

line 43-45: draw FG and RB on the same canvas.
line 46-58: make clones of FG and calculate the scaling ratios between FG and RB's.
line 60-62: subtract FG clones by RB's to get the signal distribution.

line 64-83: do the same procesures to MB's.

line 85-99: create output files; write in histograms and canvas; close output files.

3. PCF_BOTH.C

Directory:"/home/runze/Analysis/KP_MS_PCF/'PCF_both.C"
This is the code with the same function as **MS_both.C** but to deal with PCF histograms. The only difference in the structure is that we add one more function Drawline(). Therefore, only this fucntion is explained in this section. Read the previous section 2. MS_both.C for other parts.

line 17, 29 and 118-125: a macro to draw a 1D function **func** and a histogram **h** on the same canvas in range [a, b].
line 120-121: set x-axis range of h to [a, b] and y-axis range to [0.5, 1.5] (only for PCF purpose. change it as needed).

line 122-124: draw h and func on the same canvas c and return c.
Note: In this macro, Drawline() is only used to draw a horizontal line at y=1, i.e. f in line 29, to show a reference for the ratio (y-axis) in PCF histograms.

4. ADD.C

Directory:"/home/runze/Analysis/KP_MS_PCF/Add.C"
If root files are produced from multiple root files, this code is needed to add together same histograms from multiple root files (in this case, there 9: MB1, 5, 6, each with cen=1, 2, 3.)

*******Note:

1. There are actually two ways to merge MS and PCF histograms:
a) Add all FG's together, and same for RB's and MB's. Then do the subtraction for MS and division for PCF.
b) directly add worked histograms (signal for MS and ratio for PCF) together with the weight, 1 for MS signal and $\frac{1}{\delta^2}$ for PCF ratio, where δ is the bin error. To do this, you need to loop the step for each bin for all histograms. Since $\frac{1}{\delta_{1+2}^2} = \frac{1}{\delta_1^2} + \frac{1}{\delta_2^2}$ is the weight of the PCF histogram after combination, you need to scale the histogram back with the ratio δ_{1+2}^2.

2. In this macro, the method b) is applied, but in a wrong way: I mistakenly used the standard deviation error of the entire histogram as the δ. However, you can correct it easily based on current code structure and the **Scale()** function used in **MS_both.C**.

3. Since that the macro textbfMS_PCF.C can run all data at one time now, and that we look at each centrality seperately, this code is no longer used. It is presented here for completeness and in case for any other possible purpose.
Note End*****

line 12-16: set some constant parameters.
line 13: range for mass histograms.
line 14: range for PCF histograms.
line 15-16: numeber of bins per unit (1GeV) and number of bins to rebin.

line 18-21: set indices for root files to be read.

line 27: declare variables to store histogram errors, one for each charge-pair.
line 29-44: declare 12 histograms for 4 charge-pairs, each has (**FG - RB**) MS signal (called RB below), (**FG - MB**) MS signal (called MB below), (**FG / MB**) PCF ratio.

line 46-118: loop to add same histograms toghether from multiple root files.
line 46-47: loop through MB and then through cen.
line 49-53: read in MS and PCF root files. Indicate root file which MB and cen is read.
line 55-72: read in the 12 histograms from root file.

line 74-78: add RB and MB mass histograms respectively.
line 80-86: since the declared histograms are empty at the beginning, need to add pcf hitograms with normal weight = 1 in the first loop. Store histogram errors
line 89-92: For the rest loops: get histogram errors of both current histogram (combination) and newly read histogram (single one from currently read-in root file).
line 94-97: add two histograms together, each in weight of $\frac{1}{\delta^2}$.
line 99-100: update the error stored for the histogram after merge.
line 112-116: Now the weight of the histogram is $1/\delta 2$, need to scale it back with the coefficient δ^2.

line 120-124: print to check if the calculated (theoretical) error is close/same as the actual value of the final histogram (after merging through all root files).

line 126-137: ouput file; store histograms; close file.

5. COMPARE.C

Directory:"/home/runze/Analysis/KP_MS_PCF/Add/Compare.C"
This is the code to merge histograms of the same charge combinations, i.e. **KPPP+KMPM** and KPPM+KMPP. This is done respectively for RB MS, MB MS, and MB PCF. It also draws the PCF histograms of the two charge combinations on the same canvas.

Since this code is a combination of **MS_both.C** and **Add.C**, and no new function is introduced, this section will not repeat the same content. Please refer to the two sections above.

6. LEDNICKY'S CODE AND PCF_PT.C

Directory:"/home/runze/Analysis/KP_MS_PCF/Lednicky/Pt_Dist/"
For PCF, we are particularly interested in the structure of particle (1405), which is suggested by **J. M.M. Hall et al, Phys Rev Lett. 114(2015)132002** [1]. The paper points out that although (1405) is below the threshold of proton-kaon channel, it could affect the geometry of the PCF distribution. Prof. Lednicky wrote a set of codes (directory: ./clean/) to predict such PCF distributions. It needs the Pt distributions of the two decay daughters (K-p in this case) as input.
Note: 1. Lednicky's code needs to be run on NBL server. Ask Dr. Gang about the access. 2. The major macro we run to get theoretical result is **main_in_C.cpp**: first check main fucntion code **$: make**; Then run by make file **$: ./main**.

0.3 I ./clean/fsiw110707.f

This is the code to create prediction on PCF distribution by the paper [1].

line 1024: LL is the variable to control which two-particle pair is involved in analysis. According to line 36 in **fsiw110707.f**, LL = 16 represents same-sign K-p ($\boldsymbol{K^+ - p}$ and $\boldsymbol{K^- - \bar{p}}$) and LL = 17 represent opposite-sign K-p. In this code, we label kaon as the 1st

particle and proton as the second.

line 950 and 970: f0 (FDH) at line 950 is scattering length. d0 (RDH) at line 970 is effective radius.
Need to check if the two values are latest data.

line 1063-1066:
ICH, IQS, ISI and I3C are four controler of effect of interaction and statistics, which are explained in line 29-32. Now the first three are turned on. Usually you don't need to make any change.

0.4 I ./clean/main_in_C.cpp

This is the major macro to get theoretical PCF distribution we want. (This code is provided by Zhengqiao at BNL)
In this code, particles are created gaussianly in a box with edge of 2.1 and 2.2.
line 149: You can change the edge of the box here.

line 118-128: read in Pt (momentum in x-y plane, i.e. the plane perpendicular to the axle (z) axis of the STAR detector) distrubutions in different centralities.
line 140: the two variables LL and cen are put here just to remind of changing histograms in line 165 and 184 according to the board at line 143-147.

line 165 and 184: get random Pt from the read-in Pt distributions.
You can change the histograms used here to control which centrality and which two-particle pair you want to deal with.
Note that the first particle (labeled 11) is Kaon and the second particle (labeled 22) is Proton.

0.5 III PCF_Pt.C

PCF_Pt.C is the code to produce the Pt distributions as Lednicky's code needs. This code produces the Pt distributions of K^+, K^-, p, $\bar{p}$ in centrality = 1, 2, 3 respectively. The structure of the code is totally the same as that of **MS_PCF.C** up to the discard of BG filling. Therefore, the details will not be explained in this section and can be refered to the section **1. MS_PCF.C**.

7. PCF_COMPARE.C

Directory:"/home/runze/Analysis/KP_MS_PCF/Lednicky/PCF_Compare.C"
This is the code to draw predicted PCF by Lednicky's code, and the histogram from actual data on the same canvas (for both radius 2.1 and 2.2 in the last section). The content is similar as previous codes like **Compare.C** and thus would not be repeated.

8. DEBUG ON SPIKE

Directory:"/home/runze/Analysis/KP_Mix/Spike/"
Note: This section is optional. It's included only for completeness.

With assiatance of PhD students, I debuged out some bad runs that create spikes on histograms.
The codes and files are stored in foder **Spike**/.
The three codes **MB1_Debug.C**, **MB5_Debug.C** and **MB6_Debug.C** are the same up to different input data sets (These are written at early time during which I haeven't combined data files yet.).
Therefore, **MB6_Debug.C** is taken as the example here.
The code structure is based on the very original version of **MS_PCF.C** so only differences are explained.

0.6 MB6_Debug.C

This code is to create a tree to store momentum, Vz and RunID in a tree for debug.

line 89-90: add variables **MeanNum, MeanNum2, MeanPt, ... , MeanP2** for store in loop.
line 136-139: declare histograms for temporary store of momenta for all K's or p's in one event.
line 141-147: create a tree and set branches to store information like RunID, Vz and the mean values of momenta for K's and p's in each event.

For each event (entry):
line 212-214: Reset variables and histograms.
lin 384-385: Fill the momentum information of current particle pair into temporary histograms.
line 512-514: If this event contains K-p pairs so that the histograms aren't empty, get the mean value of momenta and fill the tree.

After thr running of this macro:
1. Open the root file in root, plot the 1D momentum histograms or Vz histograms to see where the spike(s) lies.
2. Plot 2D histogram of "RunID v.s. Momentum (P or Pt)" or "RunID v.s. PVtxz" and zoom in the spike area carefully to find RunID of the bad run(s).