Anti-Neutron Simualtion at BESIII

User's Guide [1]

Liang Liu[2], Xiaorong Zhou, and Haiping Peng

August 19, 2021

[1] Version AntiNeutronCorrectionSvc-00-00-03

[2] `liangzy@mail.ustc.edu.cn`

Contents

Preface

The anti-neutron correction service provide a anti-neutron simulation in EMC based on data-driven. Now it is available on ui server @USTC and ihep server @IHEP.

Structure of this guide

Chapter 1 gives a quick started about how to install and configure this service. Chapter 2 introduces the detials of the efficiency and CDFs files. Chapter 3 explains the usage of functions about this service.

If you have any question about this service, please contact Liang Liu (liangzy@mail.ustc.edu.cn).

1

Getting started

The AntiNeutronCorrectionSvc package relies on BOSS. Before installing this package make sure your environment is set properly.

1.1 Quick started

We provide two bash scripts to help users to install and configure this package, Configure_package.sh and Configure_selection_criteria.sh. With the two bash scripts, users can easily get start following the steps below.

Step 1. The AntiNeutronCorr package can be found in the following path.

```
/besfs5/groups/jpsi/jpsigroup/user/liul/software/AntiNeutronCorr/AntiNeutronCorr-00-00-01 (IHEP server)
/ustcfs2/BESUser/2018/lliu/software/AntiNeutronCorr  (USTC server )
```

Please copy the package to a location of your choice and make sure it has at least 500M free space. And then, you will fond the package with hierarchy.

```
+- AntiNeutronCorr/
   +- AntiNeutronCorr-00-00-01/
         +- bin/
         |  +- eff
         |  +- err
         |  +- fft
         +- Configure_package.sh
         +- DATA/
         |  +- Configure_selection_criteria.sh
         |  +- *.root
         |  +- ...
         +- inclde/
         |  +- Angle_FFT.h
```

```
        |  +- ...
        +- Makefile
        +- MC/
        |  +- Configure_selection_criteria.sh
        |  +- *.root
        |  +- ...
        +- share
        |  +-systematic_uncertainty.root
        +- src/
            +- Angle_FFT.cxx
            +- ...
```

Step 2. Change into the package directory and compile the program with make:

```
\$ cd /path/to/AntiNeutronCorr/AntiNeutronCorr-00-00-01/
\$ make ALL
```

Executing the command once is sufficient, unless the C++ header, source, or script file are updated or revised, or the package is moved.

Step 3. Install the service package AntiNeutronCorrectionSvc and set up the environment with the following command: ./Configure_package.sh [Option]. If it is your first time to install this package, you would run

```
\$ ./Configure_package.sh install
```

After that, you can use the command

```
\$ ./Configure_package.sh update
```

or

```
\$ ./Configure_package.sh remove
```

to update or remove the package AntiNeutronCorrectionSvc. Executing the command once is sufficient. Please remember to source your .tcshrc file to update your environment varibles.

Step 4. After setting up the environment, the efficiency matrix and error matrix file should be generated according to the selection requirement. The generate the data-driven file, change into the DATA directory and open the file Configure_selection_criteria.sh.

```
############### SELECTION CRIETRIA ##########################
Nbar_Energy=0.40 # double
Nbar_SecMom=0 # double
Nbar_Hits=0 # int
```

```
##################################################################
```

Nbar_Energy, Nbar_SecMom and Nbar_Hits stand for $E_{\bar{N}}$, Second Moment of $\bar{n}$ and the number of Hits of $\bar{n}$, respectively. The cut range for $E_{\bar{N}}$, Second Moment of $\bar{n}$ and the number of Hits of $\bar{n}$ are $[0.05, 2.0]$ GeV, $[0, 200]$ and $[0, 100]$.

After setting the shower selection criteria of $\bar{n}$, run

```
\$ ./Configure_selection_criteria.sh
```

If you need to do input output check, please change into MC directory and generate the efficiency_x_x_x.root and errormatrix_x_x_x.root files by using the similar operation.

1.2 How to Set Analysis Program

1.2.1 Header File

In the header file of you physics analysis program, please add

```
#include "AntiNeutronCorrectionSvc/AntiNeutronTrk.h"
#include "AntiNeutronCorrectionSvc/AntiNeutronCorrectionSvc.h"
#include "AntiNeutronCorrectionSvc/IAntiNeutronCorrectionSvc.h"

class YOUTALG: public Algorithm {
    private:
        AntiNeutronCorrectionSvc *m_nbar_svc;

}
```

1.2.2 Source File

In the source file of your physics analysis program, after the good photon selection, add the code from line 16 to line 49 in the following list.

```
1
2         //============================ shower
              =========================================
3         Vint ishower;
4         double etot = 0;
5         ishower.clear();
6         int goodgam = 0;
7         for( int i = evtRecEvent->totalCharged(); i < evtRecEvent->
              totalTracks(); i++){
8             EvtRecTrackIterator itTrk = evtRecTrkCol->begin()+i;
9
10 ...(other good photon selection)
```

```

            ishower.push_back(i);

        }
        // ////////////////////////////////////////////////////////////////////
        RecEmcShower *nbar_Trk;
        IAntiNeutronCorrectionSvc* nbar_svc;
        StatusCode sc_AntiNeutronCorrectionSvc = service("AntiNeutronCorrectionSvc", nbar_svc);
        if ( sc_AntiNeutronCorrectionSvc.isFailure() ){
            log << MSG::FATAL << "Could not load AntiNeutronCorrectionSvc!" << endreq;
            return sc_AntiNeutronCorrectionSvc;
        }
        m_nbar_svc = dynamic_cast<AntiNeutronCorrectionSvc*>(nbar_svc);
    // runNo = fabs(runNo);
        if(runNo < 0){
            m_nbar_svc->setAntiNeutronTrk(mc_p4nbar, mc_iniposinbar, mc_iniposilambar, 0);
            if(m_nbar_svc->isAntiNeutronCorrectionValid()){
                nbar_Trk = m_nbar_svc->getNbarShower();
                Hep3Vector emcpos(nbar_Trk->x(), nbar_Trk->y(), nbar_Trk->z());
                double dang = 200.;
                for(int j = 0; j < evtRecEvent->totalCharged(); j++) {
                    EvtRecTrackIterator jtTrk = evtRecTrkCol->begin() + j;
                    if(!(*jtTrk)->isExtTrackValid()) continue;
                    RecExtTrack *extTrk = (*jtTrk)->extTrack();
                    if(extTrk->emcVolumeNumber()==-1) continue;
                    Hep3Vector extpos = extTrk->emcPosition();
                    double angd = extpos.angle(emcpos);
                                if (angd < dang) {
                        dang = angd;
                    }
                }
                if(dang != 200.){
                    dang = dang * 180 / (CLHEP::pi);
                }
                if(fabs(dang) > m_gammaAngleCut){
                    ishower.push_back(evtRecEvent->totalTracks()); // means a brand new simulation of anti-neutron.
                }
            }
        }
```

```
        // ///////////////////////////////////////////////////////////////////////////
        int nshower= ishower.size();
        if(nshower < 1) return SUCCESS;
        m_nshower = nshower;
```

When you use this anaylsis algorithm to run MC (runNo < 0), this code will create a brand new $\bar{n}$ shower simulation *nbar_Trk. If you need to use the shower of $\bar{n}$ to do kinematic fit, you have to replace the default error matrix of $\bar{n}$ shower by calling setErrorMatrix().

```
        if(runNo < 0){
                if(!m_nbar_svc->isAntiNeutronCorrectionValid()) return SUCCESS;
                nbar_idx = nshower - 1;
                nbarTrk = nbar_Trk;
        }
        else {
                for (int k = 0; k < nshower; k++){
                        EvtRecTrackIterator itTrk = evtRecTrkCol->begin()+ishower[k];
                        if(!(*itTrk)->isEmcShowerValid()) continue;
                        RecEmcShower *emcTrk = (*itTrk)->emcShower();
                        double eraw = emcTrk->energy();
                        if(eraw>2.0) continue;
                        if(eraw>energy) {
                                energy = eraw;
                                nbar_idx = k;
                        }
                }
                nbarTrk = (*(evtRecTrkCol->begin()+ishower[nbar_idx]))->emcShower();
                if(nbarTrk->energy() < 0.4 ) return SUCCESS;
                m_nbar_svc->setErrorMatrix(nbarTrk);
        }
```

In the "cmt/requirements", please add

```
use AntiNeutronCorrectionSvc AntiNeutronCorrectionSvc-00-* Analysis
```

1.2.3 Add AntiNeutronCorrectionSvc in jobOption file

To use AntiNeutronCorrectionSvc, you need to add the headfile which is generated by step 4 and set the random seed for the package.

```
#include "\$ANTINEUTRONCORRECTIONSVCROOT/share/jobOption_AntiNeutronCorrection.txt"
AntiNeutronCorrectionSvc.rdmSeed = RANDOM;
```

The string RANDOM must be replaced by a variable of type `int`. It should be different in all jobOptions.

2

Package AntiNeutronCorr

AntiNeutronCorrectionSrc is a package to simulate anti-neutron performance in EMC based on Data-Driven method. Which means that the informations used in simulation, such as efficiency matrix and cumulative distribution function (CDF), should be evaluated from data. In this section, the calculation of the efficiency matrix and CDFs will be introduced and illustrated by using a example with requirement $E_{\bar{n}} > 0.4\,\mathrm{GeV}$, $SecMom_{\bar{n}} > 0\,\mathrm{cm}^2$ and $Hits_{\bar{n}} > 0$.

2.1 Files: efficiency_x_x_x.root and errormatrix_x_x_x.root

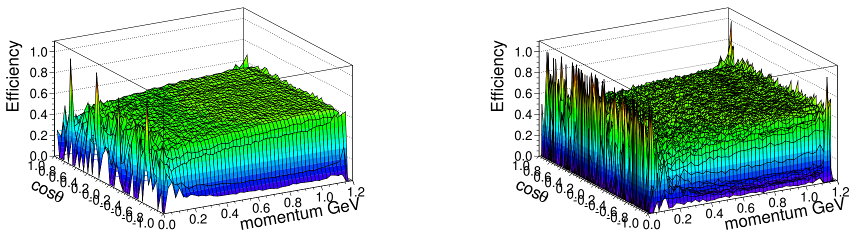
efficiency_x_x_x.root and errormatrix_x_x_x.root are create by runing ./Configure_selection_criteria.sh. Please see source file Efficiency.cxx and ErrorMatrix.cxx for details.

2.1.1 2-D efficiency matrix

The efficiency matrix of anti-neutron shower selection is calculated according to momentum vs. $\cos\theta$. There are two 2-D histograms, hratio and hratioEndCap, of type TH2D in efficiency_x_x_x.root. hratio is a 50×50 (p vs. $\cos\theta$) efficiency matrix working for $\cos\theta$ in range $[-0.72, 0.72]$, as shown in Fig. 2.1a. hratioEndCap is a 50×300 (p vs. $\cos\theta$) efficiency matrix working for $\cos\theta$ in range $[-1, 0.72]$ and $[0.72, 1]$, as shown in Fig. 2.1b.

2.1.2 Barrel and Endcap

ratio_thetaPi, i from 1 to 50, is the percentage of barrels in the whole. The data is divided into 50 bins according to momentum. A example ratio_thetaP30 is shown in Fig. 2.2.



(a) hratio (b) hratioEndCap

Figure 2.1: 2-D efficiency matrix

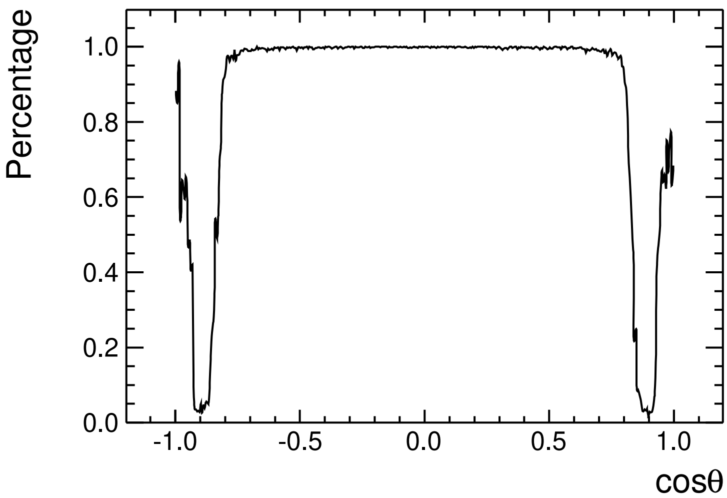
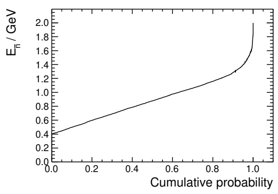


Figure 2.2: ratio_thetaP30: percentage of barrel in the whole for $\bar{n}_P$ in $[0.720, 0.744]$ GeV.

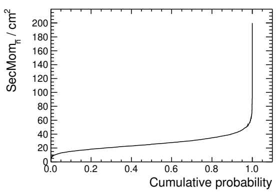
2.1.3 Cumulative Distribution Function (CDF)

nbar_energy_pi_tj, nbar_hits_pi_tj and nbar_secmom_pi_tj are CDFs of type TGraph which are evaluated from the distributions of $E_{\bar{n}}$, $Hits_{\bar{n}}$ and $SecMom_{\bar{n}}$, where p stands for momentum, t stands for $\cos\theta$, i and j from 1 to 50. An example is shown in Fig. 2.3. The data is divided into 2500 parts according to momentum and $\cos\theta$.

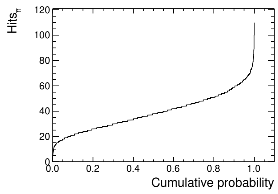
nbar_angle0_pi_tj_Ek, nbar_angle1_pi_tj_Ek, nbar_theta0_pi_tj_Ek and nbar_theta1_pi_tj_Ek are CDFs which are used to simulated the deposited spacial position of anti-neutorn, where 0 and 1 stand for barrel part and endcap parts; p, t and E stand for momentum, $\cos\theta$ and energy; i from 1 to 12, j from 1 to 20 and k from 1 to 9. The data is divided at equal intervals according to momentum and $\cos\theta$. For the energy, the step of first 6 bins is



(a) nbar_energy_p30_t25



(b) nbar_secmom_p30_t25



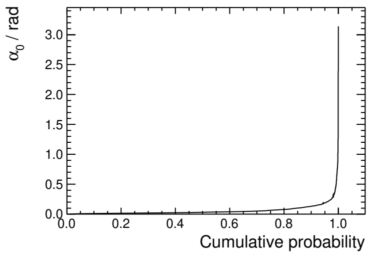
(c) nbar_hits_p30_t25

Figure 2.3: Cumulative Distribution Function of anti-neutorn shower parameters, $E_{\bar{n}}$, $SecMom_{\bar{n}}$ and $Hits_{\bar{n}}$. for $\bar{n}_P$ in $[0.720, 0.744]$ GeV and $\cos\theta$ in -0.96 to 1.00.

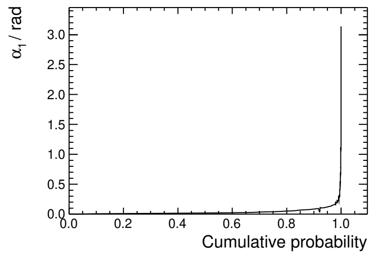
0.1 GeV; the step of bin 7 and bin 9 is 0.2 GeV; the rest is divided into one bin.

2.1.4 CDF of error matrix

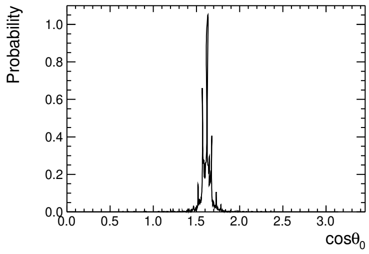
nbar_deltatheta_ei_tj and nbar_deltaphi_ei_tj are CDFs which are used to calculated the error matrix of the spacial position of anti-neutron, where i from 1 to 20, j from 1 to 50, e stand for energy, t stand for $\cos\theta$.



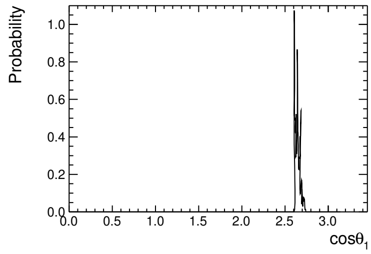
(a) nbar_angle0_p9_t10_E4



(b) nbar_angle1_p9_t2_E4

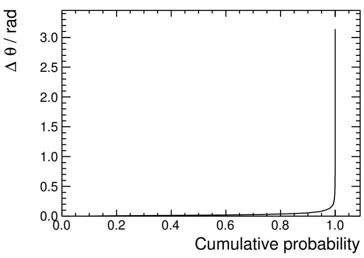


(c) nbar_theta0_p9_t10_E4

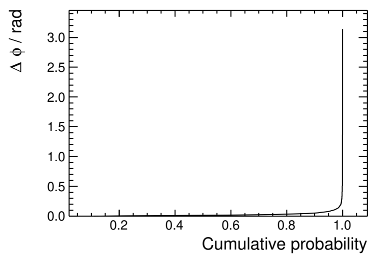


(d) nbar_theta1_p9_t2_E4

Figure 2.4: (a) and (b) are normalized CDFs of the angle between EMC shower and the direction of $p\pi^-$ recoiling system of barrel and endcap, respectively. (c) and (d) are the weight coefficients. $P_{\bar{n}}$ in range from 0.8 GeV to 0.9 GeV, $E_{\bar{n}}$ in range from 0.7 GeV to 0.8 GeV. $\cos\theta$ from -0.1 to 0.0 in barrel and from -0.9 to 0.8 in endcap.



(a) nbar_deltatheta_e10_t25



(b) nbar_deltaphi_e10_t25

Figure 2.5: (a) and (b) are normalized CDFs of the angle between EMC shower and the direction of $p\pi^-$ recoiling system of barrel and endcap, respectively. (c) and (d) are the weight coefficients. $E_{\bar{n}}$ in range from 0.9 GeV to 1.0 GeV , $\cos\theta$ from -0.04 to 0.0.

3

The Anti-neutron Correction Service

The Anti-neutron Correction Service can provide a brand new simulation of anti-neutron for MC and set the error matrix of anti-neutron for Data during the executing of algorithms. Figure 3.1 shows the structure for an example application of that service. The key idea is that a new simulation of anti-neutron of type RecEmcShower is generated and added after the original shower list.

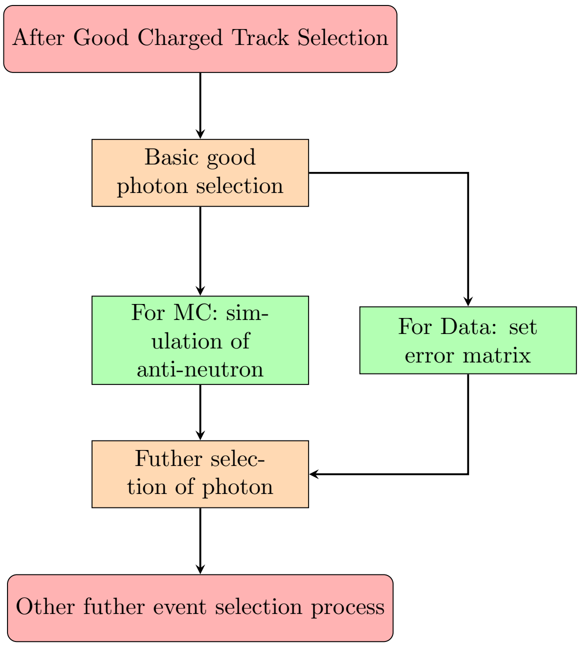


Figure 3.1: Structure of the anti-neutron correction service in algorithms.

In this chapter we describe the functions of this service and how it works in algorithms.

3.1 Functions

Four functions listed in listing 3.1 are designed to implement the simulation.

```
void *setAntiNeutronTrk(const HepLorentzVector initp4, const
    HepLorentzVector initposi, const HepLorentzVector IP, int
    statistical_uncertainty = 0);
RecEmcShower *getNbarShower();
bool isAntiNeutronCorrectionValid();
void *setErrorMatrix(RecEmcShower *nbarTrk);
```

Listing 3.1: Functions of ANTINEUTRONCORRECTIONSVC

Function setAntiNeutronTrk is used to set the information of each event. It have four parameters:

- const HepLorentzVector initp4: the truth four momentum of anti-neutron in type HepLorentzVector, getting from McParticle with initialFourMomentum().
- const HepLorentzVector initposi: the initial position of anti-neutron in MC simualtion in type HepLorentzVector, getting from McParticle with initialPosition().
- const HepLorentzVector IP: the initial position of event in type HepLorentzVector, getting from McParticle with initialPosition().
- int statistical_uncertainty = 0: set 0 to do simulation without considering the statistical uncertainty of $J/\psi \to p\bar{n}\pi^-$; set 1 to do simulation with considering the statistical uncertainty of $J/\psi \to p\bar{n}\pi^-$.

Notes:

- The usage of McParticle can be found in this table.
- If anti-neutron is a secondary particle, the initial position of anti-neutron is different from the IP of the event. If anti-neutron is a primary particle, the initial position of anti-neutron and the IP of the event is the same. For that case, user have to set the two position with the same variable of type HepLorentzVector.

The isAntiNeutronCorrectionValid() method return a variable of type bool. If the value is true, the anti-neutron can be detected under the given requirements. If the value is false, the anti-neutron can not be detected.

The getNbarShower() is used to get a anti-neutron shower of type RecEmcShower.

The setErrorMatrix(RecEmcShower *nbarTrk) is designed for running data. If users need add anti-neutron shower in the kinematic fit, the error matrix of anti-neutron shower will be replaced by a reasonable one by calling setErrorMatrix(RecEmcShower *nbarTrk).

3.2 Other remarks

- Authors suggest that users should have a specific requirment of the number of charged tracks. Do not set a requirment like at least N_{charged} tracks. Anti-neutron is a anti-matter particles. The annihilation of anti-neutron in beam pipe will cause a lot of noise tracks. It will affect the selection efficiency dramatically.

- Users are recommended to have a requirement $\alpha_{n\gamma} > 20°$, where $\alpha_{n\gamma}$ is the open angle between the direction of data-driven n simulation and the other neutral showers in EMC. Anti-neutron will have a lot of secondary showers. If users want to reconstruct other signal photons from EMC, this requirement can be used to suppress the background.