monitorPhi Cuts

Paul Simmerling

January 2021

1 PID Cuts

1.1 myElectronCutStrategies

```
// cut lvl meanings: 0 loose, 1 med, 2 tight
el_cut_strictness_lvl=["ecal_cut_lvl":1,
"nphe_cut_lvl":1,
"vz_cut_lvl":1,
"min_u_cut_lvl":1,
"min_v_cut_lvl":0,
"min_w_cut_lvl":0,
"max_u_cut_lvl":1,
"max_v_cut_lvl":0,
"max_w_cut_lvl":0,
"dcr1_cut_lvl":1,
"dcr2_cut_lvl":1,
"dcr3_cut_lvl":1,
"anti_pion_cut_lvl":1
]

def electron_from_event = new ElectronFromEvent()
electron_from_event.setElectronCutStrictness(el_cut_strictness_lvl)
electron_from_event.setElectronCutParameters(field_setting)
def myElectronCutStrategies = [
    electron_from_event.passElectronStatus,
    electron_from_event.passElectronChargeCut,
    electron_from_event.passElectronEBPIDCut,
    //on top of event  builder
    //electron_from_event.passElectronMinMomentum,
    electron_from_event.passElectronVertexCut, // sector independent but field dependent
    electron_from_event.passElectronPCALFiducialCut,
    electron_from_event.passElectronAntiPionCut, // save as rga note
    electron_from_event.passElectronEIEOCut, // technically pcal energy cut at 0.07
    electron_from_event.passElectronSamplingFractionCut,
    electron_from_event.passElectronDCFiducialCuts
]
```

1.2 myProtonCuts

```
def proton_cand = new HadronID(hadron_id:2212)
def myProtonCuts = [
    proton_cand.passHadronEBPIDCut,
    proton_cand.passTrajChi2Cut,
    proton_cand.passDCFiducialCutChi2Region1,
    proton_cand.passDCFiducialCutChi2Region2,
    proton_cand.passDCFiducialCutChi2Region3
]
```

1.3 1 ep pair

line 1152

```
def e_cand = (0..<event.npart)?.findAll{idx->event.charge[idx]<0}.collect{ii -> [ii,
    myElectronCutStrategies.collect{ el_cut -> el_cut(event,ii)} ] }.collectEntries()
def ep = e_cand.findResults{cand -> !cand.value.contains(false) ? cand.key : null
    }.findResults{ indx ->
    def ele = LorentzVector.withPID(11, event.px[indx], event.py[indx], event.pz[indx])
    return [index:indx, lv:ele, name:'el', status:event.status[indx] ]
        }.collectMany{ mele ->
        (0..<event.npart).findAll{event.pid[it]==2212 && !myProtonCuts.collect{ pr_test
            -> pr_test(event,it) }.contains(false)}.findResults{ipro->
        def pro = LorentzVector.withPID(2212,event.px[ipro], event.py[ipro],
            event.pz[ipro])
        def mpro = [index:ipro, lv:pro, name:'pro', status:event.status[ipro]]
        def ctof = (event.pid[ipro] >= 2000 && event.pid[ipro] < 4000) ? "FTOF" : "CTOF"
        def pr_fid_stat = myProtonCuts.collect{ pr_test -> pr_test(event,ipro) }
        fillProtonID(event, histos, mele.lv, pro, ipro, 'eb_'+ctof)
        return [mele,mpro]
    }
}

if( ep.size() == 1 ){....}
```

1.4 myKaonPCuts

```
def kaonP_cand = new HadronID(hadron_id:321)
def myKaonPCuts = [
    kaonP_cand.passHadronEBPIDCut,
    kaonP_cand.passTrajChi2Cut,
    kaonP_cand.passDCFiducialCutChi2Region1,
    kaonP_cand.passDCFiducialCutChi2Region2,
    kaonP_cand.passDCFiducialCutChi2Region3
]
```

1.5 myKaonMCuts

```
def kaonM_cand = new HadronID(hadron_id:-321)
def myKaonMCuts = [
    kaonM_cand.passHadronEBPIDCut,
    kaonM_cand.passTrajChi2Cut,
    kaonM_cand.passDCFiducialCutChi2Region1,
    kaonM_cand.passDCFiducialCutChi2Region2,
    kaonM_cand.passDCFiducialCutChi2Region3
]
```

1.6 At least 1 epkpkm

line 1171

```
def combs = ep.collectMany{mele,mpro->
    (0..<event.npart).findAll{ !myKaonPCuts.collect{ kp_test ->
        kp_test(event,it) }.contains(false) }.findResults{ikp->
            def kp = LorentzVector.withPID(321,event.px[ikp], event.py[ikp],
                event.pz[ikp])
            def mkp = [index:ikp, lv:kp,name:'kp',status:event.status[ikp]]
            def ctof = (event.pid[ikp] >= 2000 && event.pid[ikp] < 4000) ? "FTOF" : "CTOF"
            fillKaonPlusID(event, histos, mele.lv, kp, ikp, 'eb_'+ctof)
            def kp_fid_stat = myKaonPCuts.collect{ kp_test -> kp_test(event,ikp) }
```

```
            return [mele,mpro,mkp]
        }
    }.collectMany{mele,mpro,mkp->
        (0..<event.npart).findAll{!myKaonMCuts.collect{km_test->
        km_test(event,it)}.contains(false)}.findResults{ikm->
            def km = LorentzVector.withPID(-321,event.px[ikm], event.py[ikm],
               event.pz[ikm]) def mkm = [index:ikm, lv:km,
               name:'km',status:event.status[ikm]]
            def ctof = (event.pid[ikm] >= 2000 && event.pid[ikm] < 4000) ? "FTOF" : "CTOF"
            def km_fid_stat = myKaonMCuts.collect{ km_test -> km_test(event,ikm) }
            fillKaonMinusID(event, histos, mele.lv, km, ikm, 'eb_'+ctof)
            return [mele,mpro,mkp,mkm]
        }
    }.findResults{ele,pro,kp,km->
        return [ele,pro,kp,km]
    }
if( combs.size() > 0 ){.....}
```

2 More Event Cuts

2.1 Vz Ele Hadron Cut

```
def pass_delta_vz_el_pr = Math.abs( event.vz[el.index] - event.vz[pro.index] ) < 20
def pass_delta_vz_el_kp = Math.abs( event.vz[el.index] - event.vz[kp.index] ) < 20
def pass_delta_vz_el_km = Math.abs( event.vz[el.index] - event.vz[km.index] ) < 20
                        pass_delta_vz_el_hadron = (pass_delta_vz_el_pr &&
                           pass_delta_vz_el_kp && pass_delta_vz_el_km)
```

2.2 All in FD

```
if(el_ctof == "FTOF" && pr_ctof == "FTOF" &&
   kp_ctof == "FTOF" && km_ctof == "FTOF" ){
    all_in_fd=true
}
```

2.3 Only 1 epkpkm

```
if( combs.size() == 1 && all_in_fd && pass_delta_vz_el_hadron ){....
```

3 Exclusivity Cuts

```
def pass_epkpkmXe = phykin.epkpkmX.e() > excl_cuts.epkpkmxe[0] && phykin.epkpkmX.e() <
   excl_cuts.epkpkmxe[1]
def pass_epkpX = phykin.epkpX.mass2() > excl_cuts.epkpX[0] && phykin.epkpX.mass2() <
   excl_cuts.epkpX[1]
def pass_epkmX = phykin.epkmX.mass2() > excl_cuts.epkmX[0] && phykin.epkmX.mass2() <
   excl_cuts.epkmX[1]
def pass_ekpkmX = phykin.ekpkmX.mass2() > excl_cuts.ekpkmX[0] && phykin.ekpkmX.mass2() <
   excl_cuts.ekpkmX[1]
```

```
if( pass_epkpkmXe && pass_epkpX && pass_epkmX ){
    fillEventHistograms(event, histos, electron, proton, kaonP, kaonM, ele, pro, kp, km,
       helicity, phykin, 'pass_all_but_missing_proton')
}
else{
```

```
    fillEventHistograms(event, histos, electron, proton, kaonP, kaonM, ele, pro, kp, km,
        helicity, phykin, 'fail_at_least_missing_proton')
}
// see  missnig kp
if( pass_epkpkmXe && pass_epkpX && pass_ekpkmX  ){
    fillEventHistograms(event, histos, electron, proton, kaonP, kaonM, ele, pro, kp, km,
        helicity, phykin, 'pass_all_but_missing_kaonP')
}
else{
    fillEventHistograms(event, histos, electron, proton, kaonP, kaonM, ele, pro, kp, km,
        helicity, phykin, 'fail_at_least_missing_kaonP')
}
// see  missnig km
if( pass_epkpkmXe && pass_epkmX && pass_ekpkmX  ){
    fillEventHistograms(event, histos, electron, proton, kaonP, kaonM, ele, pro, kp, km,
        helicity, phykin, 'pass_all_but_missing_kaonM')
}
else{
    fillEventHistograms(event, histos, electron, proton, kaonP, kaonM, ele, pro, kp, km,
        helicity, phykin, 'fail_at_least_missing_kaonM')
}
// see missing energy
if(  pass_epkpX && pass_epkmX && pass_ekpkmX  ){
    fillEventHistograms(event, histos, electron, proton, kaonP, kaonM, ele, pro, kp, km,
        helicity, phykin, 'pass_all_but_missing_energy')
}
else{
    fillEventHistograms(event, histos, electron, proton, kaonP, kaonM, ele, pro, kp, km,
        helicity, phykin, 'fail_at_least_missing_energy')
}

// pass or fails only missing energy cut --> special criteria for joo
me_all_counter = me_all_counter+1
if( pass_epkpkmXe ){
    me_pass_counter = me_pass_counter + 1
    fillEventHistograms(event, histos, electron, proton, kaonP, kaonM, ele, pro, kp, km,
        helicity, phykin, 'pass_only_missing_energy')
}
else{
    me_fail_counter = me_fail_counter + 1
    fillEventHistograms(event, histos, electron, proton, kaonP, kaonM, ele, pro, kp, km,
        helicity, phykin, 'fail_only_missing_energy')
}
```

```
if( pass_epkpkmXe && pass_epkpX && pass_epkmX && pass_ekpkmX ){
```

4 Test Cuts on Background

4.1 Minimum Momentum Cut

```
def pass_el_p_min = ele.p() > excl_cuts.el_p_min
def pass_pro_p_min = pro.p() < excl_cuts.pro_p_max
def pass_kp_p_min = kp.p() < excl_cuts.kp_p_max
def pass_km_p_min = km.p() < excl_cuts.km_p_max
def pass_pt_max = phykin.pt < excl_cuts.pt_max
def pass_q2_min = phykin.q2 > excl_cuts.q2_min
def pass_w_min = phykin.w > excl_cuts.w_min
```

```
if( pass_pro_p_min ){
```

```
fillEventHistograms(event, histos, electron, proton, kaonP, kaonM, ele, pro, kp, km,
    helicity,  phykin, 'pass_all_low_pro_p')
}
if( !pass_pro_p_min ){
fillEventHistograms(event, histos, electron, proton, kaonP, kaonM, ele, pro, kp, km,
    helicity, phykin, 'pass_all_high_pro_p')
}
if( pass_kp_p_min && pass_km_p_min ){
fillEventHistograms(event, histos, electron, proton, kaonP, kaonM, ele, pro, kp, km,
    helicity, phykin, 'pass_all_low_kaonPM_p')
}
if( !pass_kp_p_min || !pass_km_p_min ){
fillEventHistograms(event, histos, electron, proton, kaonP, kaonM, ele, pro, kp, km,
    helicity, phykin, 'pass_all_high_kaonPM_p')
}
```

4.2 Delta Theta Cut

```
//delta theta cuts ( calculated - measured )
def pass_delta_theta_pro = Math.abs(phykin.delta_theta_pro) < 6
def pass_delta_theta_kp = Math.abs(phykin.delta_theta_kp) < 6
def pass_delta_theta_km = Math.abs(phykin.delta_theta_km) < 6
```

```
if( pass_delta_theta_pro ){
        fillEventHistograms(event, histos, electron, proton, kaonP, kaonM, ele, pro, kp,
            km, helicity, phykin, 'pass_all_pass_delta_theta_pro')
}
if( !pass_delta_theta_pro){
fillEventHistograms(event, histos, electron, proton, kaonP, kaonM, ele, pro, kp, km,
    helicity, phykin, 'pass_all_fail_delta_theta_pro')
}
if( pass_delta_theta_kp ){
        fillEventHistograms(event, histos, electron, proton, kaonP, kaonM, ele, pro, kp,
            km, helicity, phykin, 'pass_all_pass_delta_theta_kp')
}
if( !pass_delta_theta_kp){
fillEventHistograms(event, histos, electron, proton, kaonP, kaonM, ele, pro, kp, km,
    helicity, phykin, 'pass_all_fail_delta_theta_kp')
}
if( pass_delta_theta_km ){
        fillEventHistograms(event, histos, electron, proton, kaonP, kaonM, ele, pro, kp,
            km, helicity, phykin, 'pass_all_pass_delta_theta_km')
}
if( !pass_delta_theta_km){
fillEventHistograms(event, histos, electron, proton, kaonP, kaonM, ele, pro, kp, km,
    helicity, phykin, 'pass_all_fail_delta_theta_km')
}
if( pass_delta_theta_pro && pass_delta_theta_kp && pass_delta_theta_km ){
fillEventHistograms(event, histos, electron, proton, kaonP, kaonM, ele, pro, kp, km,
    helicity, phykin, 'pass_all_pass_delta_theta_all')
}
if( !(pass_delta_theta_pro &&pass_delta_theta_kp && pass_delta_theta_km) ){
fillEventHistograms(event, histos, electron, proton, kaonP, kaonM, ele, pro, kp, km,
    helicity, phykin, 'pass_all_fail_delta_theta_all')
}
```

4.3 Max FD Theta Cut

```
def pass_delta_theta_pro = Math.abs(phykin.delta_theta_pro) < 6
def pass_delta_theta_kp = Math.abs(phykin.delta_theta_kp) < 6
def pass_delta_theta_km = Math.abs(phykin.delta_theta_km) < 6
```

```
// limit theta to only defined FD coverage less than 35 degrees
if( pass_pro_theta_max && pass_kp_theta_max && pass_km_theta_max ){
fillEventHistograms(event, histos, electron, proton, kaonP, kaonM, ele, pro, kp, km,
    helicity, phykin, 'pass_all_pass_thetaFD')
}
else{
fillEventHistograms(event, histos, electron, proton, kaonP, kaonM, ele, pro, kp, km,
    helicity, phykin, 'pass_all_fail_thetaFD')
}
```

4.4 Chi2 Sigma Cuts

```
for( ii in 1..6){
def pass_pro_chi2 = Math.abs(event.chi2pid[ proton.index ]) < ii
def pass_kp_chi2 = Math.abs(event.chi2pid[ kaonP.index ]) < ii
def pass_km_chi2 = Math.abs(event.chi2pid[ kaonM.index ]) < ii
fillEventHistograms(event, histos, electron, proton, kaonP, kaonM, ele, pro, kp, km,
    helicity, phykin, "pass_all_raw_${ii}sigcut")
if( pass_pro_chi2 && pass_kp_chi2 && pass_km_chi2 ){
    fillEventHistograms(event, histos, electron, proton, kaonP, kaonM, ele, pro, kp, km,
        helicity, phykin, "pass_all_pass_${ii}sigcut")
}
else{
    fillEventHistograms(event, histos, electron, proton, kaonP, kaonM, ele, pro, kp, km,
        helicity, phykin, "pass_all_fail_${ii}sigcut")
}
}
```

5 Additional Cuts

5.1 Coplanarity Cut

```
def pass_cpl_pro = phykin.cplpro < excl_cuts.cpl_pro_max[1]
def pass_cpl_kp = phykin.cplkp < excl_cuts.cpl_kp_max[1]
def pass_cpl_km = phykin.cplkm < excl_cuts.cpl_km_max[1]
```

```
if( pass_cpl_pro ){
fillEventHistograms(event, histos, electron, proton, kaonP, kaonM, ele, pro, kp,
    km,helicity, phykin, 'pass_all_pass_cpl_pro')
}
if( !pass_cpl_pro ){
fillEventHistograms(event, histos, electron, proton, kaonP, kaonM, ele, pro, kp, km,
    helicity, phykin, 'pass_all_fail_cpl_pro')
}
if( pass_cpl_kp ){
fillEventHistograms(event, histos, electron, proton, kaonP, kaonM, ele, pro, kp, km,
    helicity, phykin, 'pass_all_pass_cpl_kp')
}
if( !pass_cpl_kp ){
fillEventHistograms(event, histos, electron, proton, kaonP, kaonM, ele, pro, kp, km,
    helicity, phykin, 'pass_all_fail_cpl_kp')
}
if( pass_cpl_km ){
```

```
fillEventHistograms(event, histos, electron, proton, kaonP, kaonM, ele, pro, kp, km, helicity, phykin, 'pass_all_pass_cpl_km')
}
if( !pass_cpl_km ){
fillEventHistograms(event, histos, electron, proton, kaonP, kaonM, ele, pro, kp, km, helicity, phykin, 'pass_all_fail_cpl_km')
}
if( pass_cpl_pro && pass_cpl_kp && pass_cpl_km ){
fillEventHistograms(event, histos, electron, proton, kaonP, kaonM, ele, pro, kp, km, helicity, phykin, 'pass_all_pass_cpl_all')
}
if( !(pass_cpl_pro && pass_cpl_kp && pass_cpl_km) ){
fillEventHistograms(event, histos, electron, proton, kaonP, kaonM, ele, pro, kp, km, helicity, phykin, 'pass_all_fail_cpl_all')
}
```

5.2 Delta Vz Cut

```
def pass_dvz_kpkm = (event.vz[kaonP.index] - event.vz[kaonM.index]) < excl_cuts.delta_vz_kpkm[1] && (event.vz[kaonP.index] - event.vz[kaonM.index]) > excl_cuts.delta_vz_kpkm[0]
def pass_dvz_ep = Math.abs(event.vz[electron.index] - event.vz[proton.index]) < 6
// pass just dvz kpkm check is phi mass signal improves
if( pass_dvz_kpkm ){
fillEventHistograms(event, histos, electron, proton, kaonP, kaonM, ele, pro, kp, km, helicity, phykin, "pass_all_pass_dvz_kpkm")
}
else{
fillEventHistograms(event, histos, electron, proton, kaonP, kaonM, ele, pro, kp, km, helicity, phykin, "pass_all_fail_dvz_kpkm")
}
// pass just dvz electron proton
if( pass_dvz_ep ){
fillEventHistograms(event, histos, electron, proton, kaonP, kaonM, ele, pro, kp, km, helicity, phykin, "pass_all_pass_dvz_ep")
}
else{
fillEventHistograms(event, histos, electron, proton, kaonP, kaonM, ele, pro, kp, km, helicity, phykin, "pass_all_fail_dvz_ep")
}
//passs both the delta vz cuts
if( pass_dvz_kpkm && pass_dvz_ep ){
fillEventHistograms(event, histos, electron, proton, kaonP, kaonM, ele, pro, kp, km, helicity, phykin, "pass_all_pass_dvz_both")
}
else{
fillEventHistograms(event, histos, electron, proton, kaonP, kaonM, ele, pro, kp, km, helicity, phykin, "pass_all_fail_dvz_both")
}
```

5.3 Chi2 Cut

```
// now include specific additional cuts to select final events.
def pass_pro_chi2 = Math.abs(event.chi2pid[ proton.index ]) < excl_cuts.pro_chi2[1]
def pass_kp_chi2 = Math.abs(event.chi2pid[ kaonP.index ]) < excl_cuts.kp_chi2[1]
def pass_km_chi2 = Math.abs(event.chi2pid[ kaonM.index ]) < excl_cuts.km_chi2[1]
```

5.4 Phi Mass Cut

```
if( (kp+km).mass() < excl_cuts.kpkm_mass[1] && (kp+km).mass() > excl_cuts.kpkm_mass[0] ){
fillEventHistograms(event, histos, electron, proton, kaonP, kaonM, ele, pro, kp, km,
    helicity, phykin, 'pass_all_cut_phi_mass')
fillProtonID(event, histos, ele, pro, proton.index, 'pass_all_cut_phi_mass')
fillKaonPlusID(event, histos, ele, kp, kaonP.index, 'pass_all_cut_phi_mass')
fillKaonMinusID(event, histos, ele, km, kaonM.index, 'pass_all_cut_phi_mass')
}
```

6 Phi Candidate Plots

6.1 With Additional

```
if( pass_dvz_kpkm && pass_pro_chi2 && pass_kp_chi2 && pass_km_chi2 &&
pass_cpl_pro && pass_cpl_kp && pass_cpl_km ){
```

6.2 With Phi Mass Cut

```
if( (kp+km).mass() < excl_cuts.kpkm_mass[1] && (kp+km).mass() > excl_cuts.kpkm_mass[0] ){
    reconstructed_event_present=true
    fillEventHistograms(event, histos, electron, proton, kaonP, kaonM, ele, pro, kp, km,
        helicity, phykin,
"pass_all_pass_additional_pass_phi_mass")
    fillDataResolutions( event, histos, ele, pro, kp, km, phykin,
        "pass_all_pass_additional_pass_phi_mass" )
}
}
else{
fillEventHistograms(event, histos, electron, proton, kaonP, kaonM, ele, pro, kp, km,
    helicity, phykin, "pass_all_fail_additional")
}
```

7 Phi Candidate Data

7.1 Helicity Cut

See Event Data Output

7.2 Lambda Resonance Cut

```
def pass_not_resonant1 = (pro+km).mass() < 1.5 || (pro+km).mass() > 1.58
def pass_not_resonant2 = (pro+km).mass() < 1.78 || (pro+km).mass() > 1.9
```

7.3 Event data output

```
if( t1 != null && helicity != 0 && pass_not_resonant1 && pass_not_resonant2 ) {
    writeOutEvent(t1, [ (kp+km).mass(), helicity, phykin.trentophi, phykin.q2, phykin.xb,
        phykin.t, phykin.w, event.run_number ] )
}
```

Appendices

A excl_cuts data

```
def base_path_cut = "/w/hallb-scifs17exp/clas12/bclary/CLAS12/analysis_code_fork0/projects/exclusive_phi/epkpkm_top/"
def epkpkmxe_cut = loadCuts(new File(base_path_cut+"excl_phi_me_limits_pidtype1_"+field_type+".txt"),4,4)
def ekpkmX_cut = loadCuts(new File(base_path_cut+"excl_phi_mm2_pro_limits_pidtype1_"+field_type+".txt"),4,4)
def epkmX_cut = loadCuts(new File(base_path_cut+"excl_phi_mm2_kp_limits_pidtype1_"+field_type+".txt"),4,4)
def epkpX_cut = loadCuts(new File(base_path_cut+"excl_phi_mm2_km_limits_pidtype1_"+field_type+".txt"),4,4)


def epkpkmXe_cut_range = [1 : loadCuts(new File(base_path_cut+"excl_phi_me_limits_pidtype1_"+field_type+".txt"), 1, 1),
  2 : loadCuts(new File(base_path_cut+"excl_phi_me_limits_pidtype1_"+field_type+".txt"), 2, 2),
  3 : loadCuts(new File(base_path_cut+"excl_phi_me_limits_pidtype1_"+field_type+".txt"), 3, 3),
  4 : loadCuts(new File(base_path_cut+"excl_phi_me_limits_pidtype1_"+field_type+".txt"), 4, 4)
  ]

def ekpkmX_cut_range = [1 : loadCuts(new File(base_path_cut+"excl_phi_mm2_pro_limits_pidtype1_"+field_type+".txt"), 1, 1),
2 : loadCuts(new File(base_path_cut+"excl_phi_mm2_pro_limits_pidtype1_"+field_type+".txt"), 2, 2),
3 : loadCuts(new File(base_path_cut+"excl_phi_mm2_pro_limits_pidtype1_"+field_type+".txt"), 3, 3),
4 : loadCuts(new File(base_path_cut+"excl_phi_mm2_pro_limits_pidtype1_"+field_type+".txt"), 4, 4)
                        ]

def epkmX_cut_range = [1 : loadCuts(new File(base_path_cut+"excl_phi_mm2_kp_limits_pidtype1_"+field_type+".txt"), 1, 1),
2 : loadCuts(new File(base_path_cut+"excl_phi_mm2_kp_limits_pidtype1_"+field_type+".txt"), 2, 2),
3 : loadCuts(new File(base_path_cut+"excl_phi_mm2_kp_limits_pidtype1_"+field_type+".txt"), 3, 3),
4 : loadCuts(new File(base_path_cut+"excl_phi_mm2_kp_limits_pidtype1_"+field_type+".txt"), 4, 4)
                       ]

def epkpX_cut_range = [1 : loadCuts(new File(base_path_cut+"excl_phi_mm2_km_limits_pidtype1_"+field_type+".txt"), 1, 1),
2 : loadCuts(new File(base_path_cut+"excl_phi_mm2_km_limits_pidtype1_"+field_type+".txt"), 2, 2),
3 : loadCuts(new File(base_path_cut+"excl_phi_mm2_km_limits_pidtype1_"+field_type+".txt"), 3, 3),
4 : loadCuts(new File(base_path_cut+"excl_phi_mm2_km_limits_pidtype1_"+field_type+".txt"), 4, 4)
                       ]
excl_cuts_inb = [
    epkpkmxe  : epkpkmxe_cut, //[0.08 - 0.0398*sig, 0.08 + 0.0398*sig],
    epkpX     : epkpX_cut, //[0.248 - 0.059*sig, 0.248 + 0.059*sig],
    epkmX     : epkmX_cut, //[0.248 - 0.055*sig, 0.248 + 0.055*sig],
```

```
    ekpkmX    : ekpkmX_cut, //[0.9431 - 0.0719*sig, 0.9431 + 0.0719*sig],
    el_p_min  : 0.5,
    pro_p_max : 3.5,
    kp_p_max  : 3.5,
    km_p_max  : 3.5,
    pt_max    : 0.12,
    q2_min    : 1,
    w_min     : 2,
    kpkm_mass : [1.0107, 1.0287],
    delta_vz_kpkm : [-7, 5],
    pro_chi2 : [0,6],
    kp_chi2 : [0,6],
    km_chi2 : [0,6],
    cpl_pro_max : [0, 9],
    cpl_kp_max : [0, 9],
    cpl_km_max : [0, 9]
]

excl_cuts_outb = [
    epkpkmxe  : epkpkmxe_cut,
    epkpX     : epkpX_cut,
    epkmX     : epkmX_cut,
    ekpkmX    : ekpkmX_cut,
    el_p_min  : 0.5,
    pro_p_max : 3.5,
    kp_p_max  : 3.5,
    km_p_max  : 3.5,
    pt_max    : 0.12,
    q2_min    : 1,
    w_min     : 2,
    kpkm_mass : [1.0110, 1.0281],
    delta_vz_kpkm : [-6, 9],
    pro_chi2 : [0,6],
    kp_chi2 : [0,6],
    km_chi2 : [0,6],
    cpl_pro_max : [0, 9],
    cpl_kp_max : [0, 9],
    cpl_km_max : [0, 9]
]

if( field_type == "inb" ){
    excl_cuts=excl_cuts_inb
}
else if( field_type == "outb"){
    excl_cuts=excl_cuts_outb
}
```