```
#include "myModule.h"
#include <mdst/findKs.h>
#include "n_event_printer.hh"
#include <tables/run_info.h>
#include "particle.hh"
#include <phikl_skim/B2sklSkim.h>

template <class T> inline void Erase(T& c) {
  c.erase(c.begin(), c.end());
}

extern "C" Module_descr *mdcl_myModule () {
  myModule *module = new myModule;
  Module_descr *dscr = new Module_descr ( "myModule", module );
  IpProfile::define_global(dscr);
  return dscr;
};

extern "C"{
  void rnorml_(float *,int *);
}

// ----------------------------------------------------------
void myModule::hist_def() {
  extern BelleTupleManager* BASF_Histogram;
  BelleTupleManager &tm = *BASF_Histogram;
  H[1] = tm.histogram("nev", 2, 0.0, 2.0);
  //ntp.resize(8);
  //ntp.resize(9);
  ntp.resize(6);
  string ks(" kcs kdr kdz kpx kpy kpz ksvd1 ksvd2 ");
  string pi(" ppi pisvd pie pipx pipy pipz ");
  string k(" pk lhk ksvd kpx kpy kpz ");
  string lv(" lv02 lv1d2 a3 ");
  string mylv(" mlv02 mlv1d2 ma3 ");
  string dip(" ipde cdk ");
  string common(" irun ievt size mdst eb xp tdk nd nev md mk klab iddmo ncdmo idd
iddsi mctyp1 mctyp2 mctyp3 mctyp4 mctyp5 mctyp6 mctyp7 mctyp8 mctyp9 mctyp10 mctyp11
mctyp12 mctyp13 mctyp14 mctyp15 mctyp16 mctyp17 mctyp18 mctyp19 mctyp20 mctyp21
mctyp22 mctyp23 mctyp24 mctyp25 mctyp25a mctyp26 mctyp26a mctyp27 mctyp27a nd0 ");
  string kl(" mang klx kly klz klmt klmp klmq klmf klml ecld eclt eclp ecle e9oe25
e9oe25uf ");
  //added
  string mc(" mcndst mcdpi mckpi ");
  string mcdp(" dpk dpd dppi ");
  string gpis(" igpis imgpis nmgpis ");
  string gpi(" igpi imgpi nmgpi immgpi nmmgpi immmgpi nmmmgpi ");
  string gks(" ksd1v1 ksd1v2 ksd1v3 ksd1v4 ksd1v5 ksd1v6 ksd1v7 ksd1v8 ksd1v9 ksd1v10
ksd1v11 "
         " ksd2v1 ksd2v2 ksd2v3 ksd2v4 ksd2v5 ksd2v6 ksd2v7 ksd2v8 ksd2v9 ksd2v10
ksd2v11 ");
  string gpiopp(" igpopp imgpopp nmgpopp immgpopp nmmgpopp ");
```

```
  string gpi0(" pi0d1v1 pi0d1v2 pi0d1v3 pi0d1v4 pi0d1v5 pi0d1v6 pi0d1v7 pi0d2v1
pi0d2v2 pi0d2v3 pi0d2v4 pi0d2v5 pi0d2v6 pi0d2v7 ");
  string gkl(" igklang igkl imgkl nmgkl immgkl nmmgkl immmgkl nmmmgkl immmmgkl
nmmmmgkl ");

  ntp[0] = BASF_Histogram->ntuple
    ("D0(Kspi0)",
     (common + " dm " + dip + ks + pi ).c_str());
  //(common + " dm " + dip + ks + pi + gpis + gks + gpi0 ).c_str());

  ntp[1] = BASF_Histogram->ntuple
    ("D0(Klpi0)",
     ( common + dip + kl + pi ).c_str());
  //( common + dip + kl + pi + gpis + gkl + gpi0 ).c_str());

  ntp[2] = BASF_Histogram->ntuple
    ("D0(Kspipi)",
     (common + " dm tdkst hel mkpi mpipi mkst " + dip + ks + pi ).c_str());
  //(common + " dm tdkst hel mkpi mpipi mkst " + dip + ks + pi + gpis + gks + gpi +
gpiopp ).c_str());

  ntp[3] = BASF_Histogram->ntuple
    ("D0(Klpipi)",
     (common + " tdkst hel mkpi mpipi mkst " + dip + kl + pi ).c_str());
  //(common + " tdkst hel mkpi mpipi mkst " + dip + kl + pi + gpis + gkl + gpi +
gpiopp ).c_str());

  ntp[4] = BASF_Histogram->ntuple
    ("D0(pseudoKlpi0)",
     ( common + ks + dip + pi ).c_str());
  //( common + ks + dip + pi + gpis + gks + gpi0 ).c_str());
  ntp[5] = BASF_Histogram->ntuple
    ("D0(pseudoKlpipi)",
     ( common + " tdkst hel mkpi mpipi mkst " + ks + dip + pi ).c_str());
  //( common + " tdkst hel mkpi mpipi mkst " + ks + dip + pi + gpis + gks + gpi +
gpiopp ).c_str());
};

void myModule::Clear(std::vector<Particle> & PP)
{
  Erase(PP) ;
}

// ----------------------------------------------------------
double cos(const Hep3Vector& a, const Hep3Vector& b) {
  double ptot2 = a.mag2() * b.mag2();
  return ptot2 <= 0.0 ? 0.0 : (a * b) / sqrt(ptot2);
}
// ----------------------------------------------------------
double xp(const HepLorentzVector& pSt, double ebeam2, double m2) {
  double pmax = sqrt(ebeam2 - m2);
  return pSt.vect().mag() / pmax;
```

```
}
// -------------------------------------------------------------
double lv01(const HepLorentzVector& k, const HepLorentzVector& ds1,
        double e_her, double e_ler) {
  // cos(p1, p direction in Y4S c.m.s.)
  HepLorentzVector k_boosted(k),
    v = pStar(ds1, e_her, e_ler); // in Y4S c.m.s.
  k_boosted.boost(-ds1.boostVector()); // in Ds1 c.m.s.
  return cos(k_boosted.vect(), v.vect());
}
// -------------------------------------------------------------

double kdthrust(const HepLorentzVector k, const HepLorentzVector D)
{
  // D*->D pislow, D->k pi, cos(D in lab, k in D rest)
  HepLorentzVector K(k);
  Hep3Vector bv = - D.boostVector();
  K.boost(bv); // in D c.m.s.
  return cos(K.vect(), D.vect());
}

double ksthel(const HepLorentzVector Kst, const HepLorentzVector d, const
HepLorentzVector k)
{
  // K* helicity angle
  // D*->D pislow, D->Kst piopp, Kst->K pi cos(D in Kst rest, K in Kst rest)
  HepLorentzVector K(k), D(d);
  Hep3Vector bv = - Kst.boostVector();
  K.boost(bv); D.boost(bv); // in Kst c.m.s.
  return cos(K.vect(), D.vect());
}

// -------------------------------------------------------------
void fill_d_ip_and_slow_part_info(const Hep3Vector& d, const Hep3Vector& slow,
                  const HepSymMatrix& ip_err,
                  BelleTuple* ntp) {
  // next should give the error in D & IP intersection along D direction
  // A.similarity(x) = x^T A x
  HepVector tmp; tmp = d;
  ntp->column("ipde",  float(sqrt(ip_err.similarity(tmp))));
  // angle between D and slow particle (only transverse part of the error
  // above is essential).
  ntp->column("cdk",  float(cos(d, slow)));
}
// -------------------------------------------------------------

// -------------------------------------------------------------
//added by Manmohan, 06/14/2005, calculates K0l momentum
void quadratic(HepLorentzVector p4pi,
           Particle& kl1,
           Particle& kl2,
           int& discr,
```

```
            double mD0, double mKl0,
            double& a, double& b, double& c, double& root1, double& root2)
{
  double epi = 0, piklproj = 0, M2 = 0, sqrtD = 0;
  Hep3Vector p3pi(0,0,0);
  epi = p4pi.e();
  p3pi = p4pi.vect();
  //normalize klong direction
  Hep3Vector dircos( kl1.p().vect());
  dircos *= 1.0 / dircos.mag();

  piklproj = p3pi.dot(dircos);
  M2 = (mD0*mD0 - mKl0*mKl0 - epi*epi + p3pi*p3pi);

  a = 4*(piklproj*piklproj - epi*epi);
  b = 4*M2*piklproj;
  c = (M2*M2) - 4*epi*epi*mKl0*mKl0;
  sqrtD = ((b*b - 4*a*c) > 0.) ? sqrt((b*b - 4*a*c)) : -sqrt(-(b*b - 4*a*c));

  if((b*b - 4*a*c) < 0)discr = -1;

  //solution1
  root1 = (-b - sqrtD)/(2*a);
  //HepLorentzVector p4_1(root1*(kl1.p().vect()),sqrt(mKl0*mKl0 + root1*root1));
  HepLorentzVector p4_1(root1*(dircos),sqrt(mKl0*mKl0 + root1*root1));
  //Particle newkl1(p4_1,"K0L");
  //kl1 = newkl1;
  Momentum tmp1(p4_1);
  kl1.momentum(tmp1);

  //solution2
  root2 = (-b + sqrtD)/(2*a);
  //HepLorentzVector p4_2(root2*(kl2.p().vect()),sqrt(mKl0*mKl0 + root2*root2));
  HepLorentzVector p4_2(root2*(dircos),sqrt(mKl0*mKl0 + root2*root2));
  //Particle newkl2(p4_2,"K0L");
  //kl2 = newkl2;
  Momentum tmp2(p4_2);
  kl2.momentum(tmp2);
}

//added by Manmohan for K0L reconstruction on 25 jun 2005
void recond0fromkl(std::vector<Particle> & d0_here,
            std::vector<Particle> & klnew_here,
            std::vector<Particle> & pi0new_here,
            std::vector<Particle> kl_here,
            std::vector<Particle> pi0_here,
            double mD, double mK )
{
  int k = 0;
  klnew_here.resize(kl_here.size()*pi0_here.size());
  pi0new_here.resize(kl_here.size()*pi0_here.size());
  for(std::vector<Particle>::iterator i = kl_here.begin();
```

```
      i != kl_here.end(); i++){
    for(std::vector<Particle>::iterator j = pi0_here.begin();
    j != pi0_here.end(); j++){
      //std::cout<<" no of D0 "<<k+1<<std::endl;
      int discrim = 0;
      double aa = 0;
      double bb = 0;
      double cc = 0;
      double mom1 = 0;
      double mom2 = 0;

      Particle p(*j);
      Particle p1(*i);
      Particle p2(*i);

      quadratic(p.p(), p1, p2, discrim, mD, mK, aa, bb, cc, mom1, mom2);
      if (discrim == -1)continue;

      pi0new_here.push_back(p);
      klnew_here.push_back(p1);
      HepLorentzVector p4d0 = p1.p() + p.p();
      Particle d0_cand(p4d0,"D0");
      d0_cand.relation().append(klnew_here.back());
      d0_cand.relation().append(pi0new_here.back());
      //d0_cand.relation().append(*j);
      d0_here.push_back(d0_cand);
      //d0_cand.dump("full");
      k++;
    }
  }
}


//(klpi)pi mode, overloaded recond0fromkl
void recond0fromkl(std::vector<Particle> & d0_here, int ch_here,
           std::vector<Particle> & kstnew_here,
           std::vector<Particle> & pioppnew_here,
           std::vector<Particle> & klnew_here,
           std::vector<Particle> & pinew_here,
           std::vector<Particle> piopp_here,
           std::vector<Particle> kl_here,
           std::vector<Particle> pi_here,
           double mD, double mK )
{
  int no = 0;
  kstnew_here.resize((kl_here.size()*pi_here.size()*piopp_here.size()));
  klnew_here.resize((kl_here.size()*pi_here.size()*piopp_here.size()));
  pioppnew_here.resize(kl_here.size()*pi_here.size()*piopp_here.size());
  pinew_here.resize(kl_here.size()*pi_here.size()*piopp_here.size());

  char knames[2][4] = {"K*-","K*+"};
```

```
  for(std::vector<Particle>::iterator i = kl_here.begin();
      i != kl_here.end(); i++){
    for(std::vector<Particle>::iterator j = piopp_here.begin();
    j != piopp_here.end(); j++){
      for(std::vector<Particle>::iterator k = pi_here.begin();
      k != pi_here.end(); k++){
    //std::cout<<" no of D0 "<<no+1<<std::endl;
    int discrim = 0;
    double aa = 0;
    double bb = 0;
    double cc = 0;
    double mom1 = 0;
    double mom2 = 0;

    Particle popp(*j);
    Particle p(*k);
    Particle p1(*i);
    Particle p2(*i);

    quadratic((p.p()+popp.p()), p1, p2, discrim, mD, mK, aa, bb, cc, mom1, mom2);
    if (discrim == -1)continue;

    pinew_here.push_back(p);
    klnew_here.push_back(p1);
    HepLorentzVector p4kst = p1.p() + p.p();
    Particle kst_cand(p4kst,knames[ch_here]);
    kst_cand.relation().append(klnew_here.back());
    kst_cand.relation().append(pinew_here.back());
    kstnew_here.push_back(kst_cand);
    //kst_cand.dump("full");

    if( 0.792 >= kst_cand.mass() || kst_cand.mass() >= 0.992 ) continue;

    pioppnew_here.push_back(popp);
    HepLorentzVector p4d0 = p1.p() + p.p() + popp.p();
    Particle d0_cand(p4d0,"D0");
    d0_cand.relation().append(kstnew_here.back());
    d0_cand.relation().append(pioppnew_here.back());
    d0_here.push_back(d0_cand);
    //d0_cand.dump("full");

    no++;
      }
    }
  }
}

void fill_ks_info(Particle& ks, BelleTuple* ntp, HepPoint3D ip) {
  HepPoint3D dvx(ks.mdstVee2().vx() - ip.x(),
         ks.mdstVee2().vy() - ip.y(),
         ks.mdstVee2().vz() - ip.z());
  Hep3Vector p(ks.px(), ks.py(), ks.pz());
```

```
  double dr = sqrt(dvx.x() * dvx.x() + dvx.y() * dvx.y());
  double cs = (dvx.x() * p.x() + dvx.y() * p.y())
    / dr / sqrt(p.x() * p.x() + p.y() * p.y());
  // +-10 MeV, cos>0.999
  ntp->column("kcs", float(cs));
  ntp->column("kdr", float(dr));
  ntp->column("kdz", float(dvx.z()));
  ntp->column("kpx", float(ks.p().x()));
  ntp->column("kpy", float(ks.p().y()));
  ntp->column("kpz", float(ks.p().z()));
  {
    const Mdst_trk_fit& f1 = ks.child(0).mdstCharged().trk().mhyp(2);
    const Mdst_trk_fit& f2 = ks.child(1).mdstCharged().trk().mhyp(2);
    ntp->column("ksvd1", float(f1.nhits(3) + 100*f1.nhits(4)));
    ntp->column("ksvd2", float(f2.nhits(3) + 100*f2.nhits(4)));
  }
}
void fill_pi_info(Particle& p, BelleTuple* ntp, double e_her, double e_ler) {
  ntp->column("ppi", float(pStar(p.p(), e_her, e_ler).vect().mag()));
  {
    const Mdst_trk_fit& f = p.mdstCharged().trk().mhyp(3);
    ntp->column("pisvd", float(f.nhits(3) + 100*f.nhits(4)));
  }
  ntp->column("pie", float(p.p().e()));
  ntp->column("pipx", float(p.p().x()));
  ntp->column("pipy", float(p.p().y()));
  ntp->column("pipz", float(p.p().z()));
}
void fill_k_info(Particle& p, BelleTuple* ntp, double e_her, double e_ler) {
  ntp->column("pk", float(pStar(p.p(), e_her, e_ler).vect().mag()));
  ntp->column("lhk", float(atc_pid(3,1,5,3,2).prob(&p.mdstCharged())));
  {
    const Mdst_trk_fit& f = p.mdstCharged().trk().mhyp(3);
    ntp->column("ksvd", float(f.nhits(3) + 100*f.nhits(4)));
  }
  ntp->column("kpx", float(p.p().x()));
  ntp->column("kpy", float(p.p().y()));
  ntp->column("kpz", float(p.p().z()));
}

void fill_kl_info(Particle& kl, BelleTuple* ntp){

  Mdst_klong mdstkl = kl.mdstKlong();
  Mdst_ecl ECL;
  Mdst_klm_cluster KLMC;
  if(mdstkl != 0){
    ECL = mdstkl.ecl();
    KLMC = mdstkl.klmc();
  }
  //ECL cluster info
  double distance, ecltheta, eclphi, eclenergy;
  int matchtype, ecldataquality, eclelectron;
```

```
  double e9oe25, e9oe25unf;
  if(ECL != 0){
    distance = ECL.r();
    ecltheta = ECL.theta();
    eclphi = ECL.phi();
    eclenergy = ECL.energy();
    matchtype = ECL.match();
    ecldataquality = ECL.quality();
    eclelectron = ECL.electron();
    ntp->column("ecld",distance);
    ntp->column("eclt",ecltheta);
    ntp->column("eclp",eclphi);
    ntp->column("ecle",eclenergy);
    ntp->column("e9oe25",e9oe25);
    ntp->column("e9oe25uf",e9oe25unf);
    //ntp->column("eclm",matchtype);
    //ntp->column("eclq",ecldataquality);
    //ntp->column("eclel",eclelectron);
  }

  //KLM cluster info
  double klmtheta,klmphi;
  int klmfirsthitlayer,klmquality,klmlayers;
  if(KLMC != 0){
    klmtheta = KLMC.theta();
    klmphi = KLMC.phi();
    klmfirsthitlayer = KLMC.first_layer();
    klmquality = KLMC.quality();
    klmlayers = KLMC.layers();
    ntp->column("klmt",klmtheta);
    ntp->column("klmp",klmphi);
    ntp->column("klmq",klmquality);
    ntp->column("klmf",klmfirsthitlayer);
    ntp->column("klml",klmlayers);
  }

  ntp->column("klx", cos(kl.p().vect().x()));
  ntp->column("kly", cos(kl.p().vect().y()));
  ntp->column("klz", cos(kl.p().vect().z()));

}

void fill_geninfo_pislow(BelleTuple* ntp, Particle& pis){
  if(get_hepevt(pis.mdstCharged()))
    {
      Gen_hepevt gpis = get_hepevt(pis.mdstCharged());
      int gpis_idhep = gpis.idhep();
      ntp->column("igpis",gpis_idhep);
      //cout<<"igpis = "<<gpis_idhep<<endl;

      Gen_hepevt moth_gpis = gpis.mother();
      if(moth_gpis)
```

```
    {
      int moth_gpis_idhep = moth_gpis.idhep();
      int moth_gpis_nc = moth_gpis.daLast()-moth_gpis.daFirst()+1;
      //cout<<"imgpis = "<<moth_gpis_idhep<<endl;
      ntp->column("imgpis",moth_gpis_idhep);
      ntp->column("nmgpis",moth_gpis_nc);
    }
    }
}

void fill_geninfo_pi(BelleTuple* ntp, Particle& pi){
  if(get_hepevt(pi.mdstCharged()))
    {
      Gen_hepevt gpi = get_hepevt(pi.mdstCharged());
      int gpi_idhep = gpi.idhep();
      ntp->column("igpi",gpi_idhep);
      //cout<<"igpi = "<<gpi_idhep<<endl;

      Gen_hepevt moth_gpi = gpi.mother();
      if(moth_gpi)
    {
      int moth_gpi_idhep = moth_gpi.idhep();
      int moth_gpi_nc = moth_gpi.daLast()-moth_gpi.daFirst()+1;
      //cout<<"imgpi = "<<moth_gpi_idhep<<endl;
      ntp->column("imgpi",moth_gpi_idhep);
      ntp->column("nmgpi",moth_gpi_nc);

      Gen_hepevt moth_moth_gpi = moth_gpi.mother();
      if(moth_moth_gpi)
        {
          int moth_moth_gpi_idhep = moth_moth_gpi.idhep();
          int moth_moth_gpi_nc = moth_moth_gpi.daLast()-moth_moth_gpi.daFirst()+1;
          //cout<<"immgpi = "<<moth_moth_gpi_idhep<<endl;
          ntp->column("immgpi",moth_moth_gpi_idhep);
          ntp->column("nmmgpi",moth_moth_gpi_nc);

          Gen_hepevt moth_moth_moth_gpi = moth_moth_gpi.mother();
          if(moth_moth_moth_gpi)
        {
          int moth_moth_moth_gpi_idhep = moth_moth_moth_gpi.idhep();
          int moth_moth_moth_gpi_nc = moth_moth_moth_gpi.daLast()-moth_moth_moth_gpi.
daFirst()+1;
          //cout<<"immmgpi = "<<moth_moth_moth_gpi_idhep<<endl;
          ntp->column("immmgpi",moth_moth_moth_gpi_idhep);
          ntp->column("nmmmgpi",moth_moth_moth_gpi_nc);

        }
        }
    }
    }
}
```

```
void fill_geninfo_ks(BelleTuple* ntp, Particle& ks){

  if(get_hepevt(ks.child(0).mdstCharged()))
    {
      Gen_hepevt gksd1 = get_hepevt(ks.child(0).mdstCharged());
      int gksd1_idhep = gksd1.idhep();
      ntp->column("ksd1v1",gksd1_idhep);
      //cout<<"ksd1v1  = "<<gksd1_idhep<<endl;

      Gen_hepevt moth_gksd1 = gksd1.mother();
      if(moth_gksd1)
    {
      int moth_gksd1_idhep = moth_gksd1.idhep();
      int moth_gksd1_nc = moth_gksd1.daLast()-moth_gksd1.daFirst()+1;
      //cout<<"ksd1v2  = "<<moth_gksd1_idhep<<endl;
      ntp->column("ksd1v2",moth_gksd1_idhep);
      ntp->column("ksd1v3",moth_gksd1_nc);

      Gen_hepevt moth_moth_gksd1 = moth_gksd1.mother();
      if(moth_moth_gksd1)
        {
          int moth_moth_gksd1_idhep = moth_moth_gksd1.idhep();
          int moth_moth_gksd1_nc = moth_moth_gksd1.daLast()-moth_moth_gksd1.daFirst()
+1;
          //cout<<"ksd1v4  = "<<moth_moth_gksd1_idhep<<endl;
          ntp->column("ksd1v4",moth_moth_gksd1_idhep);
          ntp->column("ksd1v5",moth_moth_gksd1_nc);

          Gen_hepevt moth_moth_moth_gksd1 = moth_moth_gksd1.mother();
          if(moth_moth_moth_gksd1)
        {
          int moth_moth_moth_gksd1_idhep = moth_moth_moth_gksd1.idhep();
          int moth_moth_moth_gksd1_nc = moth_moth_moth_gksd1.daLast()-
moth_moth_moth_gksd1.daFirst()+1;
          //cout<<"ksd1v6  = "<<moth_moth_moth_gksd1_idhep<<endl;
          ntp->column("ksd1v6",moth_moth_moth_gksd1_idhep);
          ntp->column("ksd1v7",moth_moth_moth_gksd1_nc);

          Gen_hepevt moth_moth_moth_moth_gksd1 = moth_moth_moth_gksd1.mother();
          if(moth_moth_moth_moth_gksd1)
            {
              int moth_moth_moth_moth_gksd1_idhep = moth_moth_moth_moth_gksd1.idhep()
;
              int moth_moth_moth_moth_gksd1_nc = moth_moth_moth_moth_gksd1.daLast()-
moth_moth_moth_moth_gksd1.daFirst()+1;
              //cout<<"ksd1v8  = "<<moth_moth_moth_moth_gksd1_idhep<<endl;
              ntp->column("ksd1v8",moth_moth_moth_moth_gksd1_idhep);
              ntp->column("ksd1v9",moth_moth_moth_moth_gksd1_nc);

              Gen_hepevt moth_moth_moth_moth_moth_gksd1 = moth_moth_moth_moth_gksd1.
mother();
              if(moth_moth_moth_moth_moth_gksd1)
```

```
            {
              int moth_moth_moth_moth_moth_gksd1_idhep =
moth_moth_moth_moth_moth_gksd1.idhep();
              int moth_moth_moth_moth_moth_gksd1_nc = moth_moth_moth_moth_moth_gksd1.
daLast()-moth_moth_moth_moth_moth_gksd1.daFirst()+1;
              //cout<<"ksd1v10  = "<<moth_moth_moth_moth_moth_gksd1_idhep<<endl;
              ntp->column("ksd1v10",moth_moth_moth_moth_moth_gksd1_idhep);
              ntp->column("ksd1v11",moth_moth_moth_moth_moth_gksd1_nc);
            }
            }
        }
        }
    }
    }

  if(get_hepevt(ks.child(1).mdstCharged()))
    {
      Gen_hepevt gksd2 = get_hepevt(ks.child(1).mdstCharged());
      int gksd2_idhep = gksd2.idhep();
      ntp->column("ksd2v1",gksd2_idhep);

      Gen_hepevt moth_gksd2 = gksd2.mother();
      if(moth_gksd2)
    {
      int moth_gksd2_idhep = moth_gksd2.idhep();
      int moth_gksd2_nc = moth_gksd2.daLast()-moth_gksd2.daFirst()+1;
      ntp->column("ksd2v2",moth_gksd2_idhep);
      ntp->column("ksd2v3",moth_gksd2_nc);

      Gen_hepevt moth_moth_gksd2 = moth_gksd2.mother();
      if(moth_moth_gksd2)
        {
          int moth_moth_gksd2_idhep = moth_moth_gksd2.idhep();
          int moth_moth_gksd2_nc = moth_moth_gksd2.daLast()-moth_moth_gksd2.daFirst()
+1;
          ntp->column("ksd2v4",moth_moth_gksd2_idhep);
          ntp->column("ksd2v5",moth_moth_gksd2_nc);

          Gen_hepevt moth_moth_moth_gksd2 = moth_moth_gksd2.mother();
          if(moth_moth_moth_gksd2)
        {
          int moth_moth_moth_gksd2_idhep = moth_moth_moth_gksd2.idhep();
          int moth_moth_moth_gksd2_nc = moth_moth_moth_gksd2.daLast()-
moth_moth_moth_gksd2.daFirst()+1;
          ntp->column("ksd2v6",moth_moth_moth_gksd2_idhep);
          ntp->column("ksd2v7",moth_moth_moth_gksd2_nc);

          Gen_hepevt moth_moth_moth_moth_gksd2 = moth_moth_moth_gksd2.mother();
          if(moth_moth_moth_moth_gksd2)
            {
              int moth_moth_moth_moth_gksd2_idhep = moth_moth_moth_moth_gksd2.idhep()
;
```

```
              int moth_moth_moth_moth_gksd2_nc = moth_moth_moth_moth_gksd2.daLast()-
moth_moth_moth_moth_gksd2.daFirst()+1;
              ntp->column("ksd2v8",moth_moth_moth_moth_gksd2_idhep);
              ntp->column("ksd2v9",moth_moth_moth_moth_gksd2_nc);

              Gen_hepevt moth_moth_moth_moth_moth_gksd2 = moth_moth_moth_moth_gksd2.
mother();
              if(moth_moth_moth_moth_moth_gksd2)
            {
              int moth_moth_moth_moth_moth_gksd2_idhep =
moth_moth_moth_moth_moth_gksd2.idhep();
              int moth_moth_moth_moth_moth_gksd2_nc = moth_moth_moth_moth_moth_gksd2.
daLast()-moth_moth_moth_moth_moth_gksd2.daFirst()+1;
              ntp->column("ksd2v10",moth_moth_moth_moth_moth_gksd2_idhep);
              ntp->column("ksd2v11",moth_moth_moth_moth_moth_gksd2_nc);
            }
            }
        }
        }
    }
    }
}


void fill_geninfo_pi0(BelleTuple* ntp, Particle& pi0){

  if(get_hepevt(pi0.child(0).mdstGamma()))
    {
      Gen_hepevt gpi0d1 = get_hepevt(pi0.child(0).mdstGamma());
      int gpi0d1_idhep = gpi0d1.idhep();
      //cout<<"pi0d1v1  = "<<gpi0d1_idhep<<endl;
      ntp->column("pi0d1v1",gpi0d1_idhep);


      Gen_hepevt moth_gpi0d1 = gpi0d1.mother();
      if(moth_gpi0d1)
    {
      int moth_gpi0d1_idhep = moth_gpi0d1.idhep();
      int moth_gpi0d1_nc = moth_gpi0d1.daLast()-moth_gpi0d1.daFirst()+1;
      //cout<<"pi0d1v2  = "<<moth_gpi0d1_idhep<<endl;
      ntp->column("pi0d1v2",moth_gpi0d1_idhep);
      ntp->column("pi0d1v3",moth_gpi0d1_nc);

      Gen_hepevt moth_moth_gpi0d1 = moth_gpi0d1.mother();
      if(moth_moth_gpi0d1)
        {
          int moth_moth_gpi0d1_idhep = moth_moth_gpi0d1.idhep();
          int moth_moth_gpi0d1_nc = moth_moth_gpi0d1.daLast()-moth_moth_gpi0d1.
daFirst()+1;
          //cout<<"pi0d1v4  = "<<moth_moth_gpi0d1_idhep<<endl;
          ntp->column("pi0d1v4",moth_moth_gpi0d1_idhep);
          ntp->column("pi0d1v5",moth_moth_gpi0d1_nc);
```

```
          Gen_hepevt moth_moth_moth_gpi0d1 = moth_moth_gpi0d1.mother();
          if(moth_moth_moth_gpi0d1)
        {
          int moth_moth_moth_gpi0d1_idhep = moth_moth_moth_gpi0d1.idhep();
          int moth_moth_moth_gpi0d1_nc = moth_moth_moth_gpi0d1.daLast()-
moth_moth_moth_gpi0d1.daFirst()+1;
          //cout<<"pi0d1v6  = "<<moth_moth_moth_gpi0d1_idhep<<endl;
          ntp->column("pi0d1v6",moth_moth_moth_gpi0d1_idhep);
          ntp->column("pi0d1v7",moth_moth_moth_gpi0d1_nc);
        }
        }
    }
    }

  if(get_hepevt(pi0.child(1).mdstGamma()))
    {
      Gen_hepevt gpi0d2 = get_hepevt(pi0.child(1).mdstGamma());
      int gpi0d2_idhep = gpi0d2.idhep();
      ntp->column("pi0d2v1",gpi0d2_idhep);

      Gen_hepevt moth_gpi0d2 = gpi0d2.mother();
      if(moth_gpi0d2)
    {
      int moth_gpi0d2_idhep = moth_gpi0d2.idhep();
      int moth_gpi0d2_nc = moth_gpi0d2.daLast()-moth_gpi0d2.daFirst()+1;
      ntp->column("pi0d2v2",moth_gpi0d2_idhep);
      ntp->column("pi0d2v3",moth_gpi0d2_nc);

      Gen_hepevt moth_moth_gpi0d2 = moth_gpi0d2.mother();
      if(moth_moth_gpi0d2)
        {
          int moth_moth_gpi0d2_idhep = moth_moth_gpi0d2.idhep();
          int moth_moth_gpi0d2_nc = moth_moth_gpi0d2.daLast()-moth_moth_gpi0d2.
daFirst()+1;
          ntp->column("pi0d2v4",moth_moth_gpi0d2_idhep);
          ntp->column("pi0d2v5",moth_moth_gpi0d2_nc);

          Gen_hepevt moth_moth_moth_gpi0d2 = moth_moth_gpi0d2.mother();
          if(moth_moth_moth_gpi0d2)
        {
          int moth_moth_moth_gpi0d2_idhep = moth_moth_moth_gpi0d2.idhep();
          int moth_moth_moth_gpi0d2_nc = moth_moth_moth_gpi0d2.daLast()-
moth_moth_moth_gpi0d2.daFirst()+1;
          ntp->column("pi0d2v6",moth_moth_moth_gpi0d2_idhep);
          ntp->column("pi0d2v7",moth_moth_moth_gpi0d2_nc);

        }
        }
    }
    }
}
```

```
struct geninfo {
  //int   idmoth;
  //int   hepid;
  Hep3Vector gendir;
  int gkl_idhep;
  int moth_gkl_idhep;
  int moth_gkl_nc;
  int moth_moth_gkl_idhep;
  int moth_moth_gkl_nc;
  int moth_moth_moth_gkl_idhep;
  int moth_moth_moth_gkl_nc;
  int moth_moth_moth_moth_gkl_idhep;
  int moth_moth_moth_moth_gkl_nc;

};

vector<geninfo> klongvect;

void fill_geninfo_kl(vector<geninfo> klongvect_here, BelleTuple* ntp, Particle& reckl
){
  double anglx =4.;
  //int hepid = 0;
  //int idmoth = 0;

  double gkl_idhep;
  double moth_gkl_idhep;
  double moth_gkl_nc;
  double moth_moth_gkl_idhep;
  double moth_moth_gkl_nc;
  double moth_moth_moth_gkl_idhep;
  double moth_moth_moth_gkl_nc;
  double moth_moth_moth_moth_gkl_idhep;
  double moth_moth_moth_moth_gkl_nc;


  Hep3Vector tmp3v(reckl.p().vect());
  //cout<<"reckl dir "<<reckl.p().vect()<<endl;
  //cout<<"klongvect size when accessed  "<<klongvect_here.size()<<endl;
  for (int i =0; i<klongvect_here.size(); i++) {
    Hep3Vector gen3v(klongvect_here[i].gendir);
    double angle = tmp3v.angle(gen3v);
    //cout<<"genparticle dir "<<klongvect_here[i].gendir<<endl;
    //cout<<"Angle(reckl,genkl) angle, anglx  "<<angle<<" ,  "<<anglx<<endl;

    //if (angle <anglx) { anglx = angle; hepid =klongvect_here[i].hepid; idmoth=
klongvect_here[i].idmoth;}
    if (angle <anglx) { anglx = angle;
    gkl_idhep   = klongvect_here[i].gkl_idhep;
    moth_gkl_idhep   = klongvect_here[i].moth_gkl_idhep;
    moth_gkl_nc   = klongvect_here[i].moth_gkl_nc;
```

```
    moth_moth_gkl_idhep   = klongvect_here[i].moth_moth_gkl_idhep;
    moth_moth_gkl_nc   = klongvect_here[i].moth_moth_gkl_nc;
    moth_moth_moth_gkl_idhep   = klongvect_here[i].moth_moth_moth_gkl_idhep;
    moth_moth_moth_gkl_nc   = klongvect_here[i].moth_moth_moth_gkl_nc;
    moth_moth_moth_moth_gkl_idhep   = klongvect_here[i].moth_moth_moth_moth_gkl_idhep
;
    moth_moth_moth_moth_gkl_nc   = klongvect_here[i].moth_moth_moth_moth_gkl_nc;
    }
    //ntp->column("igkl",hepid);
    ntp->column("igklang",anglx);
    //ntp->column("imgkl",idmoth);
    //ntp->column("nmkl",);
    ntp->column("igkl",gkl_idhep);
    ntp->column("imgkl",moth_gkl_idhep);
    ntp->column("nmgkl",moth_gkl_nc);
    ntp->column("immgkl",moth_moth_gkl_idhep);
    ntp->column("nmmgkl",moth_moth_gkl_nc);
    ntp->column("immmgkl",moth_moth_moth_gkl_idhep);
    ntp->column("nmmmgkl",moth_moth_moth_gkl_nc);
    ntp->column("immmmgkl",moth_moth_moth_moth_gkl_idhep);
    ntp->column("nmmmmgkl",moth_moth_moth_moth_gkl_nc);
  }
}

void get_klong_geninfo(Gen_hepevt kl, geninfo &geninfo_here){

  //geninfo_here.hepid = (*i).idhep();
  //if ((*i).mo(0)!=0) geninfo_here.idmoth = hepevt[(*i).mo(0)-1].idhep();
  geninfo_here.gendir = Hep3Vector((kl).PX(),(kl).PY(), (kl).PZ());

  Gen_hepevt gkl = kl;
  geninfo_here.gkl_idhep = gkl.idhep();

  Gen_hepevt moth_gkl = gkl.mother();
  if(moth_gkl)
    {
      geninfo_here.moth_gkl_idhep = moth_gkl.idhep();
      geninfo_here.moth_gkl_nc = moth_gkl.daLast()-moth_gkl.daFirst()+1;

      Gen_hepevt moth_moth_gkl = moth_gkl.mother();
      if(moth_moth_gkl)
    {
      geninfo_here.moth_moth_gkl_idhep = moth_moth_gkl.idhep();
      geninfo_here.moth_moth_gkl_nc = moth_moth_gkl.daLast()-moth_moth_gkl.daFirst()+
1;

      Gen_hepevt moth_moth_moth_gkl = moth_moth_gkl.mother();
      if(moth_moth_moth_gkl)
        {
          geninfo_here.moth_moth_moth_gkl_idhep = moth_moth_moth_gkl.idhep();
          geninfo_here.moth_moth_moth_gkl_nc = moth_moth_moth_gkl.daLast()-
moth_moth_moth_gkl.daFirst()+1;
```

```
          Gen_hepevt moth_moth_moth_moth_gkl = moth_moth_moth_gkl.mother();
          if(moth_moth_moth_moth_gkl)
        {
          geninfo_here.moth_moth_moth_moth_gkl_idhep = moth_moth_moth_moth_gkl.idhep
();
          geninfo_here.moth_moth_moth_moth_gkl_nc = moth_moth_moth_moth_gkl.daLast()-
moth_moth_moth_moth_gkl.daFirst()+1;

        }
        }
    }
    }
}

void fill_geninfo_piopp(BelleTuple* ntp, Particle& popp){
  if(get_hepevt(popp.mdstCharged()))
    {
      Gen_hepevt gpopp = get_hepevt(popp.mdstCharged());
      int gpopp_idhep = gpopp.idhep();
      ntp->column("igpopp",gpopp_idhep);

      Gen_hepevt moth_gpopp = gpopp.mother();
      if(moth_gpopp)
    {
      int moth_gpopp_idhep = moth_gpopp.idhep();
      int moth_gpopp_nc = moth_gpopp.daLast()-moth_gpopp.daFirst()+1;
      ntp->column("imgpopp",moth_gpopp_idhep);
      ntp->column("nmgpopp",moth_gpopp_nc);

      Gen_hepevt moth_moth_gpopp = moth_gpopp.mother();
      if(moth_moth_gpopp)
        {
          int moth_moth_gpopp_idhep = moth_moth_gpopp.idhep();
          int moth_moth_gpopp_nc = moth_moth_gpopp.daLast()-moth_moth_gpopp.daFirst()
+1;
          ntp->column("immgpopp",moth_moth_gpopp_idhep);
          ntp->column("nmmgpopp",moth_moth_gpopp_nc);
        }
    }
    }
}


// ************************************************************

// ************************************************************

void myModule::event ( BelleEvent* evptr, int* status ) {
  static NEventPrinter nev(100, cout);
  ++nev;
  *status=0;
```

```
H[1]->accumulate(0.5);

//testing coding for filling run info
Belle_event_Manager&  EveMgr =  Belle_event_Manager::get_manager();
Belle_event& Evt = *EveMgr.begin();
Evtno = Evt.EvtNo();
Runno = Evt.RunNo();
Expno = Evt.ExpNo();

Mdst_charged_Manager &charged_mng = Mdst_charged_Manager::get_manager();
eventIp     = IpProfile::position(1); // non-nil arg means event dependent
eventIp_err = IpProfile::position_err(1); // IP profile (zero means run-dep.)
// ------------------------------ Beam energy ------------------------------
double benergy = 0;
{
  Beam_energy_Manager& beam_mgr = Beam_energy_Manager::get_manager();
  Beam_energy_Manager::iterator first = beam_mgr.begin();
  if ( first == beam_mgr.end() ) benergy = 5.29; // for MC
  //cerr << "There is no Beam Energy data.\n";
  else {
    if ( first->run_flag() == 1 ) //1-on res.; 2-continumm; 3-scan
  benergy = Benergy();
    else benergy = first->E_beam();
  }
}
double benergy2 = benergy * benergy;
e_her = benergy / 5.29 * 7.998213;
e_ler = benergy / 5.29 * 3.499218;
const double theta = 0.022;


// ------------------------------ MC truth ------------------------------


Gen_hepevt_Manager& hepevt = Gen_hepevt_Manager::get_manager();
geninfo tmpgeninfo;

klongvect.erase(klongvect.begin(),klongvect.end());

for (vector<Gen_hepevt>::iterator i = hepevt.begin();
     i != hepevt.end(); ++i) {
  if ((*i).idhep()==130 || (*i).idhep()==22 || abs((*i).idhep())==2112) {
    //cout<<"genhep particle info id   "<<(*i).idhep()<<endl;
    get_klong_geninfo(*i, tmpgeninfo);
    //tmpgeninfo.hepid = (*i).idhep();
    //if ((*i).mo(0)!=0) tmpgeninfo.idmoth = hepevt[(*i).mo(0)-1].idhep();
    //tmpgeninfo.gendir = Hep3Vector((*i).PX(),(*i).PY(), (*i).PZ());
    klongvect.push_back(tmpgeninfo);
  }
}

// M C   T R U T H
```

```
  int id_d_moth = 0;
  int nc_d_moth = 0;
  int id_d_sis = 0;
  int id_d = 0;
  int mctyp1 = 0 ;
  int mctyp2 = 0 ;
  int mctyp3 = 0 ;
  int mctyp4 = 0 ;
  int mctyp5 = 0 ;
  int mctyp6 = 0 ;
  int mctyp7 = 0 ;
  int mctyp8 = 0 ;
  int mctyp9 = 0 ;
  int mctyp10 = 0 ;
  int mctyp11 = 0 ;
  int mctyp12 = 0 ;
  int mctyp13 = 0 ;
  int mctyp14 = 0 ;
  int mctyp15 = 0 ;
  int mctyp16 = 0 ;
  int mctyp17 = 0 ;
  int mctyp18 = 0 ;
  int mctyp19 = 0;
  int mctyp20 = 0;
  int mctyp21 = 0;
  int mctyp22 = 0;
  int mctyp23 = 0;
  int mctyp24 = 0;
  int mctyp25 = 0;
  int mctyp25a = 0;
  int mctyp26 = 0;
  int mctyp26a = 0;
  int mctyp27 = 0;
  int mctyp27a = 0;
  int nd0 = 0;

  Gen_hepevt* pmc_dst;
  //Make sure D0 in the generator
  for (vector<Gen_hepevt>::iterator ih = hepevt.begin();
       ih != hepevt.end(); ++ih) {

    if (abs((*ih).idhep()) == 421){
      ++nd0;

      if((*ih).mother()){
    int da1 = (*ih).mother().daFirst()-1;
    int da2 = (*ih).mother().daLast()-1;
    nc_d_moth = da2-da1+1;
    id_d_moth = (*ih).mother().idhep();
    Gen_hepevt& mc_dsis = (abs(hepevt[da1].idhep()) == 421) ? hepevt[da2] : hepevt
[da1];
    id_d_sis = mc_dsis.idhep();
```

```
      }

      // if(abs(hepevt[da1].idhep()) == 421 || abs(hepevt[da2].idhep()) == 421 )
      //    {
      //
      //  Gen_hepevt& mc_d = (abs(hepevt[da1].idhep()) == 421) ? hepevt[da1] : hepevt
[da2];
      //  id_d = mc_d.idhep();
      id_d = (*ih).idhep();
      //int da3 = mc_d.daFirst()-1;
      //int da4 = mc_d.daLast()-1;
      //int da1st = mc_d.daFirst()-1;
      //int dalast = mc_d.daLast()-1;

      int da3 = (*ih).daFirst()-1;
      int da4 = (*ih).daLast()-1;
      int da1st = (*ih).daFirst()-1;
      int dalast = (*ih).daLast()-1;

      int nc_d = da4-da3+1;

      // A n a l y s i s   m o d e s

      if((abs(hepevt[da3].idhep()) == 311 && abs(hepevt[da4].idhep()) == 111 && nc_d
== 2) ||
     (abs(hepevt[da4].idhep()) == 311 && abs(hepevt[da3].idhep()) == 111 && nc_d == 2
))
    {

      Gen_hepevt& mc_k = (abs(hepevt[da3].idhep()) == 311) ? hepevt[da3] : hepevt[da4
];

      int da5 = mc_k.daFirst()-1;
      int da6 = mc_k.daLast()-1;
      int nc_k = da6-da5+1;

      if((abs(hepevt[da5].idhep()) == 310 && nc_k == 1) ||
         (abs(hepevt[da6].idhep()) == 310 && nc_k == 1))
        {
          mctyp1 = 1;
        }

      if((abs(hepevt[da5].idhep()) == 130 && nc_k == 1) ||
         (abs(hepevt[da6].idhep()) == 130 && nc_k == 1))
        {
          mctyp2 = 1;
        }
    }

      // B a c k g r o u n d   m o d e ,   D 0 - > K 0 s   K 0 s ,   K 0 s   K 0 l
      if((abs(hepevt[da3].idhep()) == 311 && abs(hepevt[da4].idhep()) == 311 && nc_d
== 2))
```

```
    {
      Gen_hepevt& mc_k1 = hepevt[da3];
      Gen_hepevt& mc_k2 = hepevt[da4];

      int da5 = mc_k1.daFirst()-1;
      int da6 = mc_k1.daLast()-1;
      int nc_k1 = da6-da5+1;

      int da5a = mc_k2.daFirst()-1;
      int da6a = mc_k2.daLast()-1;
      int nc_k2 = da6a-da5a+1;


      if(abs(hepevt[da5].idhep()) == 310 && nc_k1 == 1 && abs(hepevt[da6].idhep()) ==
310
         && abs(hepevt[da5a].idhep()) == 310 && nc_k2 == 1 && abs(hepevt[da6a].idhep
()) == 310)
        {
          mctyp26 = 1;
        }

      if((abs(hepevt[da5].idhep()) == 130 && abs(hepevt[da6].idhep()) == 130 && nc_k1
== 1
          && abs(hepevt[da5a].idhep()) == 310 && abs(hepevt[da6a].idhep()) == 310 &&
nc_k2 == 1) ||
         (abs(hepevt[da5].idhep()) == 310 && abs(hepevt[da6].idhep()) == 310 && nc_k1
== 1
          && abs(hepevt[da5a].idhep()) == 130 && abs(hepevt[da6a].idhep()) == 130 &&
nc_k2 == 1))
        {
          mctyp26a = 1;
        }
    }

      // B a c k g r o u n d   m o d e ,   D 0 - > p i + p i - p i 0

      if((hepevt[da1st].idhep() ==211 && hepevt[dalast-1].idhep() == -211
      && hepevt[dalast].idhep() == 111 && nc_d == 3) ||

     (hepevt[da1st].idhep() ==-211 && hepevt[dalast-1].idhep() == 211
      && hepevt[dalast].idhep() == 111 && nc_d == 3) ||

     (hepevt[da1st].idhep() ==211 && hepevt[dalast-1].idhep() == 111
      && hepevt[dalast].idhep() == -211 && nc_d == 3) ||

     (hepevt[da1st].idhep() ==-211 && hepevt[dalast-1].idhep() == 111
      && hepevt[dalast].idhep() == 211 && nc_d == 3) ||

     (hepevt[da1st].idhep() ==111 && hepevt[dalast-1].idhep() == 211
      && hepevt[dalast].idhep() == -211 && nc_d == 3) ||

     (hepevt[da1st].idhep() ==111 && hepevt[dalast-1].idhep() == -211
```

```
       && hepevt[dalast].idhep() == 211 && nc_d == 3))

     {
       mctyp19 = 1;
     }

       // B a c k g r o u n d   m o d e ,   D 0 - > p i + p i - K0s/K0l

       if((hepevt[da1st].idhep() ==211 && hepevt[dalast-1].idhep() == -211
       && abs(hepevt[dalast].idhep()) == 311 && nc_d == 3) ||

      (hepevt[da1st].idhep() ==-211 && hepevt[dalast-1].idhep() == 211
       && abs(hepevt[dalast].idhep()) == 311 && nc_d == 3) ||

      (hepevt[da1st].idhep() ==211 && abs(hepevt[dalast-1].idhep()) == 311
       && hepevt[dalast].idhep() == -211 && nc_d == 3) ||

      (hepevt[da1st].idhep() ==-211 && abs(hepevt[dalast-1].idhep()) == 311
       && hepevt[dalast].idhep() == 211 && nc_d == 3) ||

      (abs(hepevt[da1st].idhep()) ==311 && hepevt[dalast-1].idhep() == 211
       && hepevt[dalast].idhep() == -211 && nc_d == 3) ||

      ((hepevt[da1st].idhep()) ==311 && hepevt[dalast-1].idhep() == -211
       && hepevt[dalast].idhep() == 211 && nc_d == 3))

     {
       int daf;
       int dal;
       int nc_K;

       if(abs(hepevt[da1st].idhep()) ==211)
         {Gen_hepevt& mc_K = (abs(hepevt[dalast].idhep()) == 311) ? hepevt[dalast] :
hepevt[dalast-1];
         daf = mc_K.daFirst()-1;dal = mc_K.daLast()-1;nc_K = dal-daf+1;}

       else if(abs(hepevt[dalast-1].idhep()) ==211)
         {Gen_hepevt& mc_K = (abs(hepevt[da1st].idhep()) == 311) ? hepevt[da1st] :
hepevt[dalast];
         daf = mc_K.daFirst()-1;dal = mc_K.daLast()-1;nc_K = dal-daf+1;}

       else if(abs(hepevt[dalast].idhep()) ==211)
         {Gen_hepevt& mc_K = (abs(hepevt[da1st].idhep()) == 311) ? hepevt[da1st] :
hepevt[dalast-1];
         daf = mc_K.daFirst()-1;dal = mc_K.daLast()-1;nc_K = dal-daf+1;}


       if(hepevt[daf].idhep() == 310 && nc_K == 1)
         {
           mctyp20 = 1;
         }
```

```
      if(hepevt[daf].idhep() == 130 && nc_K == 1)
        {
          mctyp21 = 1;
        }
    }

      // B a c k g r o u n d   m o d e ,   D 0 - > p i + K - K0s/K0l

      if((hepevt[da1st].idhep() ==211 && hepevt[dalast-1].idhep() == -321
      && (hepevt[dalast].idhep()) == 311 && nc_d == 3) ||

     (hepevt[da1st].idhep() ==211 && hepevt[dalast-1].idhep() == 311
      && (hepevt[dalast].idhep()) == -321 && nc_d == 3) ||

     (hepevt[da1st].idhep() ==311 && hepevt[dalast-1].idhep() == -321
      && hepevt[dalast].idhep() == 211 && nc_d == 3) ||

     (hepevt[da1st].idhep() ==311 && hepevt[dalast-1].idhep() == 211
      && hepevt[dalast].idhep() == -321 && nc_d == 3) ||

     (hepevt[da1st].idhep() ==-321 && hepevt[dalast-1].idhep() == 211
      && hepevt[dalast].idhep() ==311 && nc_d == 3) ||

     (hepevt[da1st].idhep() ==-321 && hepevt[dalast-1].idhep() == 311
      && hepevt[dalast].idhep() == 211 && nc_d == 3))

    {
      int daf;
      int dal;
      int nc_K;

      if((hepevt[da1st].idhep()) ==211)
        {Gen_hepevt& mc_K = ((hepevt[dalast].idhep()) == 311) ? hepevt[dalast] :
hepevt[dalast-1];
        daf = mc_K.daFirst()-1;dal = mc_K.daLast()-1;nc_K = dal-daf+1;}

      else if((hepevt[dalast-1].idhep()) ==311)
        {Gen_hepevt& mc_K = ((hepevt[dalast-1]));
        daf = mc_K.daFirst()-1;dal = mc_K.daLast()-1;nc_K = dal-daf+1;}

      else if((hepevt[dalast].idhep()) ==211)
        {Gen_hepevt& mc_K = ((hepevt[da1st].idhep()) == 311) ? hepevt[da1st] : hepevt
[dalast-1];
        daf = mc_K.daFirst()-1;dal = mc_K.daLast()-1;nc_K = dal-daf+1;}

      if(hepevt[daf].idhep() == 310 && nc_K == 1)
        {
          mctyp27 = 1;
        }
      if(hepevt[daf].idhep() == 130 && nc_K == 1)
        {
          mctyp27a = 1;
```

```
        }
    }

      // Background modes, D0 -> K0 R, R = broad
resonance.

      if((abs(hepevt[da3].idhep()) == 311 && ((abs(hepevt[da4].idhep()) == 113 ||
                          abs(hepevt[da4].idhep()) == 223 ||
                          abs(hepevt[da4].idhep()) == 221 ||
                          abs(hepevt[da4].idhep()) == 331 ||
                          abs(hepevt[da4].idhep()) == 10221 ||
                          abs(hepevt[da4].idhep()) == 225 ||
                          abs(hepevt[da4].idhep()) == 10331)) && nc_d == 2)
     ||

     (abs(hepevt[da4].idhep()) == 311 && ((abs(hepevt[da3].idhep()) == 113 ||
                          abs(hepevt[da3].idhep()) == 223 ||
                          abs(hepevt[da3].idhep()) == 221 ||
                          abs(hepevt[da3].idhep()) == 331 ||
                          abs(hepevt[da3].idhep()) == 10221 ||
                          abs(hepevt[da3].idhep()) == 225 ||
                          abs(hepevt[da3].idhep()) == 10331)) && nc_d == 2))
    {

      Gen_hepevt& mc_k = (abs(hepevt[da3].idhep()) == 311) ? hepevt[da3] : hepevt[da4
];
      Gen_hepevt& mc_R = (abs(hepevt[da3].idhep()) == 311) ? hepevt[da4] : hepevt[da3
];

      int da5 = mc_k.daFirst()-1;
      int da6 = mc_k.daLast()-1;
      int nc_k = da6-da5+1;

      if((abs(hepevt[da5].idhep()) == 310 && nc_k == 1) ||
         (abs(hepevt[da6].idhep()) == 310 && nc_k == 1))
        {
          if(mc_R.idhep() == 113){mctyp5 = 1;}
          else if(mc_R.idhep() == 223){mctyp6 = 1;}
          else if(mc_R.idhep() == 221){mctyp7 = 1;}
          else if(mc_R.idhep() == 331){mctyp8 = 1;}
          else if(mc_R.idhep() == 10221){mctyp9 = 1;}
          else if(mc_R.idhep() == 225){mctyp10 = 1;}
          else if(mc_R.idhep() == 10331){mctyp11 = 1;}
        }

      if((abs(hepevt[da5].idhep()) == 130 && nc_k == 1) ||
         (abs(hepevt[da6].idhep()) == 130 && nc_k == 1))
        {
          if(mc_R.idhep() == 113){mctyp12 = 1;}
          else if(mc_R.idhep() == 223){mctyp13 = 1;}
          else if(mc_R.idhep() == 221){mctyp14 = 1;}
          else if(mc_R.idhep() == 331){mctyp15 = 1;}
```

```
          else if(mc_R.idhep() == 10221){mctyp16 = 1;}
          else if(mc_R.idhep() == 225){mctyp17 = 1;}
          else if(mc_R.idhep() == 10331){mctyp18 = 1;}
        }

    }


      // O t h e r   B a c k g r o u n d   M o d e s , D 0 - > K -   P i + ,   K * -   K + a n d   K * +   K -

      //std::cout<<"da3   "<<hepevt[da3].idhep()<<"   da4  "<<hepevt[da4].idhep()<<" nc_d  "<<nc_d<<std::endl;

      if((abs(hepevt[da3].idhep()) == 323 && ((abs(hepevt[da4].idhep()) == 321 )) && nc_d == 2)
     ||
     (abs(hepevt[da3].idhep()) == 321 && ((abs(hepevt[da4].idhep()) == 323 )) && nc_d == 2)
     ||
     (hepevt[da3].idhep()) == -321 && (hepevt[da4].idhep()) == 211 && nc_d == 2
     ||
     (hepevt[da3].idhep()) == 321 && (hepevt[da4].idhep()) == -211 && nc_d == 2)

    {

      Gen_hepevt& mc_K = (abs(hepevt[da3].idhep()) == 321) ? hepevt[da3] : hepevt[da4];
      Gen_hepevt& mc_KstOrPi = (abs(hepevt[da3].idhep()) == 321) ? hepevt[da4] : hepevt[da3];


      if(abs(mc_K.idhep()) == 321 && abs(mc_KstOrPi.idhep()) == 211) mctyp22 = 1;
      if(mc_K.idhep() == 321 && mc_KstOrPi.idhep() == -323) mctyp23 = 1;
      if(mc_K.idhep() == -321 && mc_KstOrPi.idhep() == 323) mctyp24 = 1;

    }


      // C a l i b r a t i o n   m o d e s

      if((abs(hepevt[da3].idhep()) == 323 && abs(hepevt[da4].idhep()) == 211 && nc_d == 2) ||
     (abs(hepevt[da4].idhep()) == 323 && abs(hepevt[da3].idhep()) == 211 && nc_d == 2 ))
    {

      Gen_hepevt& mc_kst = (abs(hepevt[da3].idhep()) == 323) ? hepevt[da3] : hepevt[da4];

      int da5 = mc_kst.daFirst()-1;
      int da6 = mc_kst.daLast()-1;
```

```
      int nc_kst = da6-da5+1;


      if((abs(hepevt[da5].idhep()) == 311 && abs(hepevt[da6].idhep()) == 211 &&
nc_kst == 2) ||
         (abs(hepevt[da6].idhep()) == 311 && abs(hepevt[da5].idhep()) == 211 &&
nc_kst == 2))
        {
          Gen_hepevt& mc_k = (abs(hepevt[da5].idhep()) == 311) ? hepevt[da5] : hepevt
[da6];

          int da7 = mc_k.daFirst()-1;
          int da8 = mc_k.daLast()-1;
          int nc_k = da8-da7+1;

          if((abs(hepevt[da7].idhep()) == 310 && nc_k == 1) ||
         (abs(hepevt[da8].idhep()) == 310 && nc_k == 1))
        {
          mctyp3 = 1;
        }

          if((abs(hepevt[da7].idhep()) == 130 && nc_k == 1) ||
         (abs(hepevt[da8].idhep()) == 130 && nc_k == 1))
        {
          mctyp4 = 1;
        }

        }
    }

      // K0*- Pi+ and cc

      if((abs(hepevt[da3].idhep()) == 10321 && abs(hepevt[da4].idhep()) == 211 &&
nc_d == 2) ||
     (abs(hepevt[da4].idhep()) == 10321 && abs(hepevt[da3].idhep()) == 211 && nc_d ==
2))
    {

      Gen_hepevt& mc_k0st = (abs(hepevt[da3].idhep()) == 10321) ? hepevt[da3] :
hepevt[da4];

      int da5 = mc_k0st.daFirst()-1;
      int da6 = mc_k0st.daLast()-1;
      int nc_k0st = da6-da5+1;

      if((abs(hepevt[da5].idhep()) == 311 && abs(hepevt[da6].idhep()) == 211 &&
nc_k0st == 2) ||
         (abs(hepevt[da6].idhep()) == 311 && abs(hepevt[da5].idhep()) == 211 &&
nc_k0st == 2))
        {
          Gen_hepevt& mc_k = (abs(hepevt[da5].idhep()) == 311) ? hepevt[da5] : hepevt
[da6];
```

```
        int da7 = mc_k.daFirst()-1;
        int da8 = mc_k.daLast()-1;
        int nc_k = da8-da7+1;

        if((abs(hepevt[da7].idhep()) == 310 && nc_k == 1) ||
       (abs(hepevt[da8].idhep()) == 310 && nc_k == 1))
      {
        mctyp25 = 1;
      }

        if((abs(hepevt[da7].idhep()) == 130 && nc_k == 1) ||
       (abs(hepevt[da8].idhep()) == 130 && nc_k == 1))
      {
        mctyp25a = 1;
      }

      }
  }

  }
}

// R e c o n s t r u c t i o n  s t a r t s  h e r e .

double klthetares = 0.016456;
double klphires = 0.018358;
double ksthetares = 0.0024362;
double ksphires = 0.0018613;


//std::cout<<" before k,pi "<<std::endl;
// ------------------------------ K, pi charged -----------------------------
vector<Particle> k[2], pi[2];
for(int ch = 0; ch <=1 ; ++ch){
  Clear(k[ch]);
  Clear(pi[ch]);
}
{
  char pi_NAMES[2][4] = {"PI-", "PI+"};
  char  k_NAMES[2][3] = {"K-" , "K+" };
  for(std::vector<Mdst_charged>::iterator i = charged_mng.begin();
  i != charged_mng.end(); ++i){
    if ( !good_charged(*i) ) continue;
    if ( abs(i->trk().mhyp(2).helix(0)) > 2 ||
     abs(i->trk().mhyp(2).helix(3) - eventIp.z()) > 4 ) continue;

    double lhk = atc_pid(3,1,5,3,2).prob(&(*i));
    int ch = ((i->charge() < 0.) ? 0 : 1);
    pi[ch].push_back(Particle(*i, Ptype(pi_NAMES[ch])));
    if ( lhk > 0.1 ) k[ch].push_back(Particle(*i, Ptype(k_NAMES[ch])));
  }
```

```
  }

  //std::cout<<" before k0s "<<std::endl;
  // ------------------------------ K0S ------------------------------
  vector<Particle> ks;
  Clear(ks);
  { // taken from utility.cc: goodKs req. is added
    Mdst_vee2_Manager &veeMgr = Mdst_vee2_Manager::get_manager();
    for(std::vector<Mdst_vee2>::iterator i = veeMgr.begin();
        i != veeMgr.end(); ++i){
      FindKs fk;
      fk.candidates(*i, eventIp);
      if ( fk.goodKs() ) ks.push_back(Particle(*i));
    }
  }
  //  makeKs(ks);
  for (int i=0; i<(int)ks.size(); ++i) {
    HepPoint3D dvx(ks[i].mdstVee2().vx() - eventIp.x(),
           ks[i].mdstVee2().vy() - eventIp.y(), 0);
    Hep3Vector p(ks[i].px(), ks[i].py(), 0);
    double dr = dvx.perp();
    double cs = (dvx * p) / dvx.mag() / p.mag();
    double m = ks[i].mass();
    // +-10 MeV, cos>0.999
  }


  //pseudo-K0L s
  vector<Particle> pskl[2];
  for(int ch = 0; ch <=1 ; ++ch){
    Clear(pskl[ch]);
  }
  {
    for ( std::vector<Particle>::iterator i = ks.begin();
      i != ks.end(); ++i) {
      Particle newks(*i);
      Hep3Vector p3(newks.p3().unit());
      //std::cout<<" before smearing p3 "<<p3<<std::endl;
      int iDim = 2;
      float rndm[2];
      rnorml_(rndm,&iDim);

      double newphi = p3.phi() + (sqrt((klphires)*(klphires) - (ksphires)*(ksphires))
)*rndm[0];
      double newtheta = p3.theta() + (sqrt((klthetares)*(klthetares) - (ksthetares)*
(ksthetares)))*rndm[1];
      p3.setPhi(newphi);
      p3.setTheta(newtheta);
      //std::cout<<" after smearing p3 "<<p3<<std::endl;
      HepLorentzVector p4(p3,newks.p().e());
      newks.momentum().momentum(p4,newks.momentum().dp());
      pskl[0].push_back(newks);
```

```
    pskl[1].push_back(newks);
    //pskl[0].back().dump("full");
    //pskl[1].back().dump("full");
  }
}

//std::cout<<" before pi0 "<<std::endl;
// ------------------------------ pi0 ------------------------------
vector<Particle> pi0;
Clear(pi0);
makePi0(pi0);
for (std::vector<Particle>::iterator l = pi0.begin(); l!=pi0.end(); ++l) {
  if ( l->child(0).p().e()<0.05 ||
   l->child(1).p().e()<0.05 ) { pi0.erase(l); --l; continue; }
}

//std::cout<<" before k0l "<<std::endl;
// ------------------------------ K0L ------------------------------
vector<Particle> klong[2];
for(int ch = 0; ch <=1 ; ++ch){
  Clear(klong[ch]);
}
{
  //char kl_NAMES[2][4] = {"kl0", ""};
  Mdst_klong_Manager &klongMgr = Mdst_klong_Manager::get_manager();
  for(std::vector<Mdst_klong>::iterator i = klongMgr.begin();
  i != klongMgr.end(); i++){
    //klong[0].push_back(Particle(*i, Ptype("K0L")));
    //klong[1].push_back(Particle(*i, Ptype("K0L")));
    klong[0].push_back(Particle(*i));
    klong[1].push_back(Particle(*i));

  }
}
//std::cout<<" before kst "<<std::endl;
//------------------------------K*(+-)------------------------------
vector<Particle> kst[2];
for(int ch = 0; ch <=1 ; ++ch){
  Clear(kst[ch]);
}
{
  char knames[2][4] = {"K*-","K*+"};
  for (int ch=0; ch<2; ++ch) {
    int opp = 1 - ch;
    // K*- -> K0s pi-
    combination(kst[ch], knames[ch], ks, pi[ch], 0.100);
  }
}

//std::cout<<" before d0 "<<std::endl;

// ------------------------------ D0 ------------------------------
```

```
//new vectors from D0->K0L pi0 + cc
vector<Particle> klnew1, pi0new;
Clear(klnew1);
Clear(pi0new);
//new vectors from D0 -> K*- pi+, K*- -> K0L pi-  + cc
vector<Particle> kstnew[2], pioppnew[2], klnew2[2], pinew[2] ;
for(int ch = 0; ch <=1 ; ++ch){
  Clear(kstnew[ch]);
  Clear(pioppnew[ch]);
  Clear(klnew2[ch]);
  Clear(pinew[ch]);
}
vector<Particle> d0;
Clear(d0);
{
  // D0->K0s pi0 + cc
  combination(d0, "D0", ks, pi0, 0.200, 0.100);

  // D0->K0L pi0 + cc
  //std::cout<<" before kldo "<<std::endl;
  double mDzero = 1.8645;
  double mKlong = .497672;
  recond0fromkl(d0, klnew1, pi0new, klong[0], pi0, mDzero, mKlong);
  //std::cout<<" after kldo "<<std::endl;

  // D0 -> K*- pi+, K*- -> K0S pi-  + cc
  for (int ch=0; ch<2; ++ch) {
    int opp = 1 - ch;
    combination(d0, "D0", kst[ch], pi[opp], 0.100);
  }

  // D0 -> K*- pi+, K*- -> K0L pi-  + cc
  char knames[2][4] = {"K*-","K*+"};
  for (int ch=0; ch<2; ++ch) {
    int opp = 1 - ch;
    recond0fromkl(d0, ch, kstnew[ch], pioppnew[opp], klnew2[ch], pinew[ch],
          pi[opp], klong[1], pi[ch], mDzero, mKlong);
  }

}

//std::cout<<" before dst "<<std::endl;

// ------------------------------ D0 from pseudo K0L ------------------------------
//new vectors from D0-> pseudo K0L pi0 + cc
vector<Particle> psklnew1, pspi0new;
Clear(psklnew1);
Clear(pspi0new);

//new vectors from D0 -> K*- pi+, K*- -> pseudo K0L pi-  + cc
vector<Particle> pskstnew[2], pspioppnew[2], psklnew2[2], pspinew[2] ;
for(int ch = 0; ch <=1 ; ++ch){
```

```
      Clear(pskstnew[ch]);
      Clear(pspioppnew[ch]);
      Clear(psklnew2[ch]);
      Clear(pspinew[ch]);
    }
    vector<Particle> psd0;
    Clear(psd0);
    {
      // D0 -> pseudo K0L pi0 + cc
      //std::cout<<" before pseudo kldo "<<std::endl;
      double mDzero = 1.8645;
      double mKlong = .497672;
      recond0fromkl(psd0, psklnew1, pspi0new, pskl[0], pi0, mDzero, mKlong);
      //std::cout<<" after pseudo kldo "<<std::endl;

      // D0 -> K*- pi+, K*- -> pseudo K0L pi-  + cc
      char knames[2][4] = {"K*-","K*+"};
      for (int ch=0; ch<2; ++ch) {
        int opp = 1 - ch;
        recond0fromkl(psd0, ch, pskstnew[ch], pspioppnew[opp], psklnew2[ch], pspinew[ch
],
              pi[opp], pskl[1], pi[ch], mDzero, mKlong);
      }

    }

    // ------------------------------ D* ------------------------------
    vector<Particle> dst[2];
    for(int ch = 0; ch <=1 ; ++ch){
      Clear(dst[ch]);
    }
    {
      char names[2][4] = {"D*-", "D*+"};
      for (int ch=0; ch<2; ++ch){
        //std::cout<<" inside dst before combination "<<std::endl;
        combination(dst[ch], names[ch], d0, pi[ch], 0.200); // D*+ -> D0 pi+
        //std::cout<<" inside dst after combination "<<std::endl;
      }
    }

    H[1]->accumulate(1.5);

    // ------------------------------ D* from pseudo K0L ---------------
    vector<Particle> psdst[2];
    for(int ch = 0; ch <=1 ; ++ch){
      Clear(psdst[ch]);
    }
    {
      char names[2][4] = {"D*-", "D*+"};
      for (int ch=0; ch<2; ++ch){
        //std::cout<<" inside dst before combination "<<std::endl;
        combination(psdst[ch], names[ch], psd0, pi[ch], 0.200); // D*+ -> D0 pi+
```

```
    //std::cout<<" inside dst after combination "<<std::endl;
  }
}

int multiplicity = 0 ;

//double fw[2][3] = {0,0,0};

for (int ch=0; ch<2; ++ch) {
  //calcuFoxWofram(dst[ch],fw[ch],e_her,e_ler,theta);
  for (int i=0; i<(int)dst[ch].size(); ++i) {
    //std::cout<<" flag1a "<<std::endl;
    enum {kspi = 0, klpi , kspipi, klpipi, other} type;
    Particle& p = dst[ch][i];
    int nd;
    if      ( abs(p.child(0).child(0).pType().lund()) == 310 &&
      abs(p.child(0).child(1).pType().lund()) == 111 )
  { nd = p.child(0).nChildren(); type = kspi; }
    else if ( abs(p.child(0).child(0).pType().lund()) == 130 &&
      abs(p.child(0).child(1).pType().lund()) == 111 )
  { nd = p.child(0).nChildren(); type = klpi; }
    else if ( abs(p.child(0).child(0).child(0).pType().lund()) == 310 &&
      abs(p.child(0).child(0).child(1).pType().lund()) == 211 )
  { nd = p.child(0).nChildren();          type = kspipi; }
    else if ( abs(p.child(0).child(0).child(0).pType().lund()) == 130 &&
      abs(p.child(0).child(0).child(1).pType().lund()) == 211 )
  { nd = p.child(0).nChildren();          type = klpipi; }
    else
  { nd = p.child(0).nChildren();          type = other; }
    int iNt = int(type);
    if ( p.child(0).pType().lund() < 0 ) nd = -nd;
    //std::cout<<" flag1b "<<std::endl;
    double msq = p.p().m2();
    double m = sqrt(msq);

    //std::cout<<" type "<<type<<std::endl;
    double xP = xp(pStar(p.p(), e_her, e_ler), benergy2, msq);

    double m1 = p.p().m();
    double m2 = p.child(0).p().m();
    bool accept = false;
    if ( type == kspi && ((m1 - m2)+1.8645) < 2.03 )          accept = true;
    else if ( type == klpi && m1 < 2.03 )                     accept = true;
    else if ( type == kspipi && ((m1 - m2)+1.8645) < 2.03 )   accept = true;
    else if ( type == klpipi && m1 < 2.03 )                   accept = true;
    else if ( type == other )                                accept = false;
    if (accept == true) *status = 1;
    else continue;
    ++multiplicity;
    //std::cout<<" flag1c "<<std::endl;

    fill_d_ip_and_slow_part_info(p.child(0).p().vect(), p.child(1).p().vect(),
```

```
eventIp_err, ntp[iNt]);

      ntp[iNt]->column("nd0",  nd0);
      ntp[iNt]->column("iddmo",  id_d_moth);

      ntp[iNt]->column("iddsi",  id_d_sis);
      ntp[iNt]->column("idd",  id_d);
      ntp[iNt]->column("ncdmo",  nc_d_moth);

      ntp[iNt]->column("mctyp1",  mctyp1);
      ntp[iNt]->column("mctyp2",  mctyp2);
      ntp[iNt]->column("mctyp3",  mctyp3);
      ntp[iNt]->column("mctyp4",  mctyp4);
      ntp[iNt]->column("mctyp5",  mctyp5);
      ntp[iNt]->column("mctyp6",  mctyp6);
      ntp[iNt]->column("mctyp7",  mctyp7);
      ntp[iNt]->column("mctyp8",  mctyp8);
      ntp[iNt]->column("mctyp9",  mctyp9);
      ntp[iNt]->column("mctyp10",  mctyp10);
      ntp[iNt]->column("mctyp11",  mctyp11);
      ntp[iNt]->column("mctyp12",  mctyp12);
      ntp[iNt]->column("mctyp13",  mctyp13);
      ntp[iNt]->column("mctyp14",  mctyp14);
      ntp[iNt]->column("mctyp15",  mctyp15);
      ntp[iNt]->column("mctyp16",  mctyp16);
      ntp[iNt]->column("mctyp17",  mctyp17);
      ntp[iNt]->column("mctyp18",  mctyp18);
      ntp[iNt]->column("mctyp19",  mctyp19);
      ntp[iNt]->column("mctyp20",  mctyp20);
      ntp[iNt]->column("mctyp21",  mctyp21);
      ntp[iNt]->column("mctyp22",  mctyp22);
      ntp[iNt]->column("mctyp23",  mctyp23);
      ntp[iNt]->column("mctyp24",  mctyp24);
      ntp[iNt]->column("mctyp25",  mctyp25);
      ntp[iNt]->column("mctyp25a",  mctyp25a);
      ntp[iNt]->column("mctyp26",  mctyp26);
      ntp[iNt]->column("mctyp26a",  mctyp26a);
      ntp[iNt]->column("mctyp27",  mctyp27);
      ntp[iNt]->column("mctyp27a",  mctyp27a);

      ntp[iNt]->column("size",  multiplicity);

      //testing code for filling run info
      ntp[iNt]->column("ievt", Evtno );
      ntp[iNt]->column("irun", 100*Runno+Expno );
      //      ntp[iNt]->column("exp", Expno );

      ntp[iNt]->column("eb",  benergy);
      ntp[iNt]->column("xp",  xP     );
      ntp[iNt]->column("mdst",  m     );
      ntp[iNt]->column("nd",  nd     );
      ntp[iNt]->column("nev",  n_processed_events);
```

```
  if ( type == kspi ) { // --------------- K0s pi0 ---------------
Particle  Dst = p;
Particle  D   = Dst.child(0);
Particle  Pis  = Dst.child(1);
Particle  K   = D.child(0);
Particle  Pi = D.child(1);
//std::cout<<"  flag2  "<<std::endl;
double mk = K.mass();
double klab = K.p().vect().mag();
double dm = Dst.mass() - D.mass();
double theta_D_K = kdthrust(K.p(),D.p());
ntp[iNt]->column("tdk",  float(theta_D_K));
ntp[iNt]->column("mk",  float(mk));
ntp[iNt]->column("klab",  float(klab    ));
ntp[iNt]->column("dm", float(dm));
ntp[iNt]->column("md", float(D.mass()));
fill_ks_info(K,  ntp[iNt], eventIp);
fill_pi_info(Pis, ntp[iNt], e_her, e_ler);
/*
  fill_geninfo_pislow(ntp[iNt],Pis);
  fill_geninfo_pi0(ntp[iNt],Pi);
  fill_geninfo_ks(ntp[iNt],K);
*/

  }

  else if ( type == klpi ) { // --------------- K0L pi0---------------
Particle  Dst = p;
Particle  D   = Dst.child(0);
Particle  Pis  = Dst.child(1);
Particle  K   = D.child(0);
Particle  Pi = D.child(1);
//std::cout<<"  flag3  "<<std::endl;
double mk = K.mass();
double klab = K.p().vect().mag();
double theta_D_K = kdthrust(K.p(),D.p());

int ecl_ID = -1;
if(K.mdstKlong()){
  if(K.mdstKlong().ecl()) ecl_ID = K.mdstKlong().ecl().get_ID();
}
else if(K.mdstEcl()) ecl_ID = K.mdstEcl().get_ID();

double missingangle = cos_missing(K.p(),ecl_ID);
ntp[iNt]->column("tdk",  float(theta_D_K));
ntp[iNt]->column("mang",  missingangle);
ntp[iNt]->column("mk",  mk);
ntp[iNt]->column("klab",  klab    );
ntp[iNt]->column("md", D.mass());
fill_kl_info(K, ntp[iNt]);
fill_pi_info(Pis, ntp[iNt], e_her, e_ler);
```

```
/*
fill_geninfo_pislow(ntp[iNt],Pis);
fill_geninfo_pi0(ntp[iNt],Pi);
fill_geninfo_kl(klongvect,ntp[iNt],K);
*/
  }

  else if ( type == kspipi ) { // ---------------- K0S pi+ pi- ----------------
Particle  Dst = p;
Particle  D   = Dst.child(0);
Particle  Kst = D.child(0);
Particle  K   = Kst.child(0);
Particle  Pis  = Dst.child(1);
Particle  Piopp  = D.child(1);
Particle  Pi  = Kst.child(1);
//std::cout<<"  flag4  "<<std::endl;
double mk = K.mass();
double klab = K.p().vect().mag();
double dm = Dst.mass() - D.mass();
double theta_D_K = kdthrust(K.p(),D.p());
double theta_D_Kst = kdthrust(Kst.p(),D.p());
double ksthelicity = ksthel(Kst.p(),D.p(),K.p());
ntp[iNt]->column("tdk",  float(theta_D_K));
ntp[iNt]->column("tdkst",  float(theta_D_Kst));
ntp[iNt]->column("hel",  float(ksthelicity));
ntp[iNt]->column("mk",  float(mk));
ntp[iNt]->column("klab",  float(klab    ));
ntp[iNt]->column("mkst",  Kst.mass());
ntp[iNt]->column("dm", float(dm));
ntp[iNt]->column("md", float(D.mass()));
ntp[iNt]->column("mpipi", float((Piopp.p() + Pi.p()).mag()));
ntp[iNt]->column("mkpi", float((K.p() + Piopp.p()).mag()));
fill_ks_info (K,  ntp[iNt], eventIp);
fill_pi_info(Pis, ntp[iNt], e_her, e_ler);
/*
  fill_geninfo_pislow(ntp[iNt],Pis);
  fill_geninfo_piopp(ntp[iNt],Piopp);
  fill_geninfo_pi(ntp[iNt],Pi);
  fill_geninfo_ks(ntp[iNt],K);
*/
  }

  else if ( type == klpipi ) { // -------------- K0L pi+ pi- ----------------
Particle  Dst = p;
Particle  D   = Dst.child(0);
Particle  Kst = D.child(0);
Particle  K   = Kst.child(0);
Particle  Pis = Dst.child(1);
Particle  Piopp  = D.child(1);
Particle  Pi  = Kst.child(1);
double mkst = Kst.mass();
double mk = K.mass();
```

```
    double klab = K.p().vect().mag();
    double theta_D_K = kdthrust(K.p(),D.p());
    double theta_D_Kst = kdthrust(Kst.p(),D.p());
    double ksthelicity = ksthel(Kst.p(),D.p(),K.p());

    int ecl_ID = -1;
    if(K.mdstKlong()){
      if(K.mdstKlong().ecl()) ecl_ID = K.mdstKlong().ecl().get_ID();
    }
    else if(K.mdstEcl()) ecl_ID = K.mdstEcl().get_ID();

    double missingangle = cos_missing(K.p(),ecl_ID);
    ntp[iNt]->column("tdk",  float(theta_D_K));
    ntp[iNt]->column("tdkst",  float(theta_D_Kst));
    ntp[iNt]->column("hel",  float(ksthelicity));
    ntp[iNt]->column("mang",  missingangle);
    ntp[iNt]->column("mkst",  float(mkst));
    ntp[iNt]->column("mk",  float(mk));
    ntp[iNt]->column("klab",  float(klab    ));
    ntp[iNt]->column("md", float(D.mass()));
    ntp[iNt]->column("mpipi", float((Piopp.p() + Pi.p()).mag()));
    ntp[iNt]->column("mkpi", float((K.p() + Piopp.p()).mag()));
    fill_kl_info (K,  ntp[iNt]);
    fill_pi_info(Pis, ntp[iNt], e_her, e_ler);

    /*
      fill_geninfo_pislow(ntp[iNt],Pis);
      fill_geninfo_piopp(ntp[iNt],Piopp);
      fill_geninfo_pi(ntp[iNt],Pi);
      fill_geninfo_kl(klongvect,ntp[iNt],K);
    */
      }

      double e = pStar(p.p(), e_her, e_ler).e();
      ntp[iNt]->dumpData();
    }
  }

  H[1]->accumulate(1.5);

  int multiplicity_here = 0;

  //for pseudo K0L modes
  for (int ch=0; ch<2; ++ch) {
    for (int i=0; i<(int)psdst[ch].size(); ++i) {
      //std::cout<<" flag1a "<<std::endl;
      enum {psklpi = 0, psklpipi} type;
      Particle& p = psdst[ch][i];
      int nd;
      if      ( abs(p.child(0).child(0).pType().lund()) == 310 &&
        abs(p.child(0).child(1).pType().lund()) == 111 )
    { nd = p.child(0).nChildren(); type = psklpi; }
```

```
      else if ( abs(p.child(0).child(0).child(0).pType().lund()) == 310 &&
        abs(p.child(0).child(0).child(1).pType().lund()) == 211 )
    { nd = p.child(0).nChildren();          type = psklpipi; }
      int iNt = int(type) + 4;
      if ( p.child(0).pType().lund() < 0 ) nd = -nd;
      //std::cout<<" flag1b "<<std::endl;
      double msq = p.p().m2();
      double m = sqrt(msq);

      //std::cout<<" type "<<type<<std::endl;
      double xP = xp(pStar(p.p(), e_her, e_ler), benergy2, msq);
      double m1 = p.p().m();
      //double m2 = p.child(0).p().m();
      bool accept = false;
      if ( type == psklpi && m1 < 2.03 )                  accept = true;
      else if ( type == psklpipi && m1 < 2.03 )            accept = true;

      if (accept == false) continue;
      //std::cout<<" flag1c "<<std::endl;

      ++multiplicity_here;

      fill_d_ip_and_slow_part_info(p.child(0).p().vect(), p.child(1).p().vect(),
eventIp_err, ntp[iNt]);

      ntp[iNt]->column("nd0", nd0);
      ntp[iNt]->column("iddmo",  id_d_moth);

      ntp[iNt]->column("iddsi",  id_d_sis);
      ntp[iNt]->column("idd",  id_d);
      ntp[iNt]->column("ncdmo",  nc_d_moth);

      ntp[iNt]->column("mctyp1",  mctyp1);
      ntp[iNt]->column("mctyp2",  mctyp2);
      ntp[iNt]->column("mctyp3",  mctyp3);
      ntp[iNt]->column("mctyp4",  mctyp4);
      ntp[iNt]->column("mctyp5",  mctyp5);
      ntp[iNt]->column("mctyp6",  mctyp6);
      ntp[iNt]->column("mctyp7",  mctyp7);
      ntp[iNt]->column("mctyp8",  mctyp8);
      ntp[iNt]->column("mctyp9",  mctyp9);
      ntp[iNt]->column("mctyp10",  mctyp10);
      ntp[iNt]->column("mctyp11",  mctyp11);
      ntp[iNt]->column("mctyp12",  mctyp12);
      ntp[iNt]->column("mctyp13",  mctyp13);
      ntp[iNt]->column("mctyp14",  mctyp14);
      ntp[iNt]->column("mctyp15",  mctyp15);
      ntp[iNt]->column("mctyp16",  mctyp16);
      ntp[iNt]->column("mctyp17",  mctyp17);
      ntp[iNt]->column("mctyp18",  mctyp18);
      ntp[iNt]->column("mctyp19",  mctyp19);
      ntp[iNt]->column("mctyp20",  mctyp20);
```

```
  ntp[iNt]->column("mctyp21",  mctyp21);
  ntp[iNt]->column("mctyp22",  mctyp22);
  ntp[iNt]->column("mctyp23",  mctyp23);
  ntp[iNt]->column("mctyp24",  mctyp24);
  ntp[iNt]->column("mctyp25",  mctyp25);
  ntp[iNt]->column("mctyp25a",  mctyp25a);
  ntp[iNt]->column("mctyp26",  mctyp26);
  ntp[iNt]->column("mctyp26a",  mctyp26a);
  ntp[iNt]->column("mctyp27",  mctyp27);
  ntp[iNt]->column("mctyp27a",  mctyp27a);

  ntp[iNt]->column("size", multiplicity_here);

  ntp[iNt]->column("ievt", Evtno );
  ntp[iNt]->column("irun", 100*Runno+Expno );
  //ntp[iNt]->column("exp", Expno );

  ntp[iNt]->column("eb",  benergy);
  ntp[iNt]->column("xp",  xP     );
  ntp[iNt]->column("mdst",  m     );
  ntp[iNt]->column("nd",  nd     );
  ntp[iNt]->column("nev",  n_processed_events);

  if ( type == psklpi ) { // ---------------pseudo K0L pi0---------------
Particle  Dst = p;
Particle  D   = Dst.child(0);
Particle  Pis  = Dst.child(1);
Particle  K   = D.child(0);
Particle  Pi = D.child(1);
//std::cout<<"  flag3  "<<std::endl;
double mk = K.mass();
double klab = K.p().vect().mag();
double theta_D_K = kdthrust(K.p(),D.p());

int ecl_ID = -1;
//      if(K.mdstKlong()){
//    if(K.mdstKlong().ecl()) ecl_ID = K.mdstKlong().ecl().get_ID();
//  }
//  else if(K.mdstEcl()) ecl_ID = K.mdstEcl().get_ID();

double missingangle = cos_missing(K.p(),ecl_ID);
ntp[iNt]->column("tdk",  float(theta_D_K));
ntp[iNt]->column("mk",  mk);
ntp[iNt]->column("klab",  klab    );
ntp[iNt]->column("md", D.mass());
//fill_kl_info(K, ntp[iNt]);
fill_ks_info(K, ntp[iNt], eventIp);
fill_pi_info(Pis, ntp[iNt], e_her, e_ler);
//newly added, 12 oct 2006
/*
  fill_geninfo_pislow(ntp[iNt],Pis);
  fill_geninfo_pi0(ntp[iNt],Pi);
```

```
      fill_geninfo_ks(ntp[iNt],K);
    */
      }
      else if ( type == psklpipi ) { // --------------pseudo  K0L pi+ pi- -----------
----
    Particle  Dst = p;
    Particle  D   = Dst.child(0);
    Particle  Kst = D.child(0);
    Particle  K   = Kst.child(0);
    Particle  Pis  = Dst.child(1);
    Particle  Piopp  = D.child(1);
    Particle  Pi  = Kst.child(1);
    double mkst = Kst.mass();
    double mk = K.mass();
    double klab = K.p().vect().mag();
    double theta_D_K = kdthrust(K.p(),D.p());
    double theta_D_Kst = kdthrust(Kst.p(),D.p());
    double ksthelicity = ksthel(Kst.p(),D.p(),K.p());

    int ecl_ID = -1;
    //  if(K.mdstKlong()){
    //    if(K.mdstKlong().ecl()) ecl_ID = K.mdstKlong().ecl().get_ID();
    //  }
    //  else if(K.mdstEcl()) ecl_ID = K.mdstEcl().get_ID();

    HepLorentzVector p4_kl = K.p();
    double missingangle = cos_missing(p4_kl,ecl_ID);
    ntp[iNt]->column("tdk",  float(theta_D_K));
    ntp[iNt]->column("tdkst",  float(theta_D_Kst));
    ntp[iNt]->column("hel",  float(ksthelicity));
    ntp[iNt]->column("mkst",  float(mkst));
    ntp[iNt]->column("mk",  float(mk));
    ntp[iNt]->column("klab",  float(klab    ));
    ntp[iNt]->column("md", float(D.mass()));
    ntp[iNt]->column("mpipi", float((Piopp.p() + Pi.p()).mag()));
    ntp[iNt]->column("mkpi", float((K.p() + Piopp.p()).mag()));
    //fill_kl_info (K,  ntp[iNt]);
    fill_ks_info(K, ntp[iNt], eventIp);
    fill_pi_info(Pis, ntp[iNt], e_her, e_ler);
    //newly added, 12 oct 2006
    /*
      fill_geninfo_pislow(ntp[iNt],Pis);
      fill_geninfo_piopp(ntp[iNt],Piopp);
      fill_geninfo_pi(ntp[iNt],Pi);
      fill_geninfo_ks(ntp[iNt],K);
    */
      }

      ntp[iNt]->dumpData();
    }
  }
  //std::cout<<" events processed  "<<n_processed_events<<std::endl;
```

```
  ++n_processed_events;
}

HepLorentzVector myModule ::Y4S_momentum(void)
{
  static const double theta = 0.022;
  double Eler  = 3.5;
  double Eher  = 7.9965;
  return HepLorentzVector( Eher*sin(theta), 0.0, Eher*cos(theta)-Eler, Eher+Eler );
}

void myModule :: calMissingMomentum(void)
{
  double px=0;
  double py=0;
  double pz=0;
  Mdst_charged_Manager& chtrk = Mdst_charged_Manager::get_manager();
  chtrk.begin();

  for(int i=0; i<(int)chtrk.size(); i++){
    px += chtrk[i].px();
    py += chtrk[i].py();
    pz += chtrk[i].pz();
  }

  Mdst_gamma_Manager& gamtrk = Mdst_gamma_Manager::get_manager();
  gamtrk.begin();
  for(int i=0; i<(int)gamtrk.size(); i++){
    px += gamtrk[i].px();
    py += gamtrk[i].py();
    pz += gamtrk[i].pz();
  }

  HepLorentzVector p4_Y4S = Y4S_momentum();
  px = p4_Y4S.vect().x() - px;
  py = p4_Y4S.vect().y() - py;
  pz = p4_Y4S.vect().z() - pz;
  /*
  pmiss_x = px;
  pmiss_y = py;
  pmiss_z = pz;
  pmiss_rho = sqrt(px*px+py*py+pz*pz);
  */
  _pmiss_x = px;
  _pmiss_y = py;
  _pmiss_z = pz;
  _pmiss_rho = sqrt(px*px+py*py+pz*pz);

}
double myModule :: cos_missing(HepLorentzVector p4_kl, int ecl_ID)
{
  if(ecl_ID==-1)
```

```
    return (
            ( p4_kl.vect().x()*_pmiss_x + p4_kl.vect().y()*_pmiss_y + p4_kl.vect().z
()*_pmiss_z )
            / ( p4_kl.vect().mag()*_pmiss_rho )
            );

  double pmiss_x = _pmiss_x;
  double pmiss_y = _pmiss_y;
  double pmiss_z = _pmiss_z;
  double pmiss_rho = _pmiss_rho;
  Mdst_gamma_Manager& gamtrk = Mdst_gamma_Manager::get_manager();
  gamtrk.begin();
  for(int i=0; i<(int)gamtrk.size(); i++){
    if( gamtrk[i].ecl().get_ID() != ecl_ID ) continue;
    pmiss_x += gamtrk[i].px();
    pmiss_y += gamtrk[i].py();
    pmiss_z += gamtrk[i].pz();
  }
  pmiss_rho = sqrt(pmiss_x*pmiss_x+pmiss_y*pmiss_y+pmiss_z*pmiss_z);
  return (
          ( p4_kl.vect().x()*pmiss_x + p4_kl.vect().y()*pmiss_y + p4_kl.vect().z()*
pmiss_z
)
          / ( p4_kl.vect().mag()*pmiss_rho )
          );
}
```