```
#ifndef Align_hh
#define Align_hh

#include "TVectorD.h"
#include "TMatrixD.h"
#include "TFormula.h"
#include "TH1.h"
#include "TFile.h"
#include "TLinearFitter.h"
#include "TObject.h"
#include "TClonesArray.h"
#include "Trace.hh"


class Align:public TObject{

public:

   TLinearFitter *fitxz;
   TLinearFitter *fityz;
   Double_t *xp;
   Double_t *yp;
   Double_t *zp;
   Double_t *e;

  int npts,ntrace;
  float ntrace_eff[2][6];
  int nfit1, nfit2, nfit2_2, nfit3, nfit4;
  int nfit5, nfit6, nfit7, nfit8;
  float xval1[5000];
  float yval1[5000];
  float xval2[5000];
  float yval2[5000];
  float xval2_2[5000];
  float yval2_2[5000];
  float xval3[5000];
  float yval3[5000];
  float xval4[5000];
  float yval4[5000];
  float xval5[5000];
  float yval5[5000];
  float xval6[5000];
  float yval6[5000];
  float xval7[5000];
  float yval7[5000];
  float xval8[5000];
  float yval8[5000];
  float depz;
  float resx[6],resex[6];
  float resy[6],resey[6];
  float reseffx[6],reseffy[6];
  float resxech[4],resexech[4];
  float resyech[4],reseyech[4];
  float ressx[4],ressy[4]; // residus pour scan va
  float resx_pos0[2],resy_pos0[2];
  float resx_pos4[2],resy_pos4[2];
  float ppiste_n[6];
  int deux_plaq_n[6];
  double xdx[6],ydy[6]; // coordonnees x/y avant les corrections dxy et dyx
  double xpospred[6];
  double ypospred[6];
  int ztdr[6]; // ordre des tdrs par rapport a la direction du faisceau
  int ref1,ref2; // detecteurs reference pour l'alignement - position dans la le
cture
  int ind3,ind4; // les deux autres echelles AMS02
  int indp1,indp2; // les deux petits detecteurs

  int mauvais[6][1024]; // signalisation des canaux chauds
```



```
  int align; // pour la gestion de l'alignement
  /*  align = 0 rien fait
            = 1 determine la position du faisceau -> algpar.dat
            = 2 <- algpar.dat, dxy et dxy entre detecteurs ref. -> algpar.dat
            = 3 <- algpar.dat, dxx et dyy entre detecteurs non-ref -> algpar.dat

            = 4 <- algpar.dat, dxy et dyx entre detecteurs non-ref -> algpar.dat

            = 5 <- algpar.dat, mis-a-jour de la position du faisceau -> algpar.d
at
  */

  double algpar[15][6];
  double algpar_2[15][6]; // si une deuxieme plaquette
  /* parametres d'alignement
     0 - decalage en x (centre de la distribution du faisceau a 0 cm)
     1 - decalage en y
     2 - pente dxx
     3 - intersection dxx
     4 - pente dyy
     5 - intersection dyy
     6 - pente dxy
     7 - intersection dxy
     8 - pente dyx
     9 - intersection dyx
    10 - angle dxdz
    11 - le centre de la rotation dans le plan xz
    12 - angle dydz
    13 - le centre de la rotation dans le plan yz
    14 - position nominale en z */

  /* alignement 0 scan 50296
     alignement 1 run 60067
     alignement 2 run 60625
     alignement 3 run 60592 (a)*/
  double algpar_fich[8][11][6];
  /* deplacement en y entre les deux senseurs run 1237 */
  double delta_y_ind4;

  /* dimensions x et y des plaquettes aux silicium (cm) */
  float plaq_dim[2];
  float plaq_dim_active[2];
  /* l'ecart nominal entre deux plaquettes (cm) */
  float ecart;

  /* parametres pour les amas: maximum nombre des pistes utitlise pour la positi
on
     des amas */
  int amaspar[2][6];

  /* fonctions eta */
  TH1D *feta_p[6],*feta_n[6];
  /* fonctions eta avec trois pistes */
  TH1D *feta3_p[6],*feta3_n[6];
  float bord_eta3_p[6],bord_eta3_n[6];

  //! Default contructor

  Align(int ival, int run);
  //! Default destructor
  ~Align(){};
  // mise-a-jour des centres du faisceau
  void MaJXYpos(int mode,double x00,double y00,double x01,double y01,double x02,
double y02,double x03,double y03,
                double x04, double y04, double x05, double y05) {
  // en cm
    if (mode==0) {
      algpar[0][0] = x00;
      algpar[1][0] = y00;
```

```
      algpar[0][1] = x01;
      algpar[1][1] = y01;
      algpar[0][2] = x02;
      algpar[1][2] = y02;
      algpar[0][3] = x03;
      if (deux_plaq()) algpar[1][3]+=y03;
      else algpar[1][3] = y03;
      algpar[0][4] = x04;
      algpar[1][4] = y04;
      algpar[0][5] = x05;
      algpar[1][5] = y05;
    }
    if (mode==1) {
      algpar[0][0]+=x00;
      algpar[1][0]+=y00;
      algpar[0][1]+=x01;
      algpar[1][1]+=y01;
      algpar[0][2]+=x02;
      algpar[1][2]+=y02;
      algpar[0][3]+=x03;
      algpar[1][3]+=y03;
      algpar[0][4]+= x04;
      algpar[1][4]+= y04;
      algpar[0][5]+= x05;
      algpar[1][5]+= y05;
    }
    if (deux_plaq()) {
      algpar_2[0][3] = algpar[0][3];
      if (mode==0) algpar_2[1][3]=y03;
      else if (mode==1) algpar_2[1][3]+=y03;
    }
    return;
}
  // mise-a-jour de la correction pour l'angle entre les axes des detecteurs ref
1 et ref2
  void MaJRotRef1(double pente1,double inter1, double pente2, double inter2) {
    algpar[6][ref1] = pente1;
    algpar[7][ref1] = inter1;
    algpar[8][ref1] = pente2;
    algpar[9][ref1] = inter2;
    return;
  }
  // mise-a-jour de la correction pour l'angle entre les axes parelleles des det
ecteurs 3 et 4
  void MaJRotxxyy(double pente1,double inter1, double pente2, double inter2,
                  double pente3,double inter3, double pente4, double inte
r4,
                  double pente5,double inter5, double pente6, double inter6,
    double pente7,double inter7, double pente8, double inter8, double pente2_2,
double inter2_2) {
    algpar[2][ind3] = pente1;
    algpar[3][ind3] = inter1;
    algpar[4][ind3] = pente2;
    algpar[5][ind3] = inter2;
    algpar_2[4][ind3] = pente2_2;
    algpar_2[5][ind3] = inter2_2;
    algpar[2][ind4] = pente3;
    algpar[3][ind4] = inter3;
    algpar[4][ind4] = pente4;
    algpar[5][ind4] = inter4;
    algpar[2][indp1] = pente5;
    algpar[3][indp1] = inter5;
    algpar[4][indp1] = pente6;
    algpar[5][indp1] = inter6;
    algpar[2][indp2] = pente7;
    algpar[3][indp2] = inter7;
    algpar[4][indp2] = pente8;
    algpar[5][indp2] = inter8;
    return;
```



```
  }
  // mise-a-jour de la correction pour l'angle entre les axes dans le plan xy po
ur 3 et 4
  void MaJRotxyyx(double pente1,double inter1, double pente2, double inter2,
                  double pente3,double inter3, double pente4, double inte
r4,
                  double pente5,double inter5, double pente6, double inter6,
                  double pente7, double inter7, double pente8, double int
er8) {
    algpar[6][ind3] = pente1;
    algpar[7][ind3] = inter1;
    algpar[8][ind3] = pente2;
    algpar[9][ind3] = inter2;
    algpar[6][ind4] = pente3;
    algpar[7][ind4] = inter3;
    algpar[8][ind4] = pente4;
    algpar[9][ind4] = inter4;
    algpar[6][indp1] = pente5;
    algpar[7][indp1] = inter5;
    algpar[8][indp1] = pente6;
    algpar[9][indp1] = inter6;
    algpar[6][indp2] = pente7;
    algpar[7][indp2] = inter7;
    algpar[8][indp2] = pente8;
    algpar[9][indp2] = inter8;
    return;
  }

  // si besoin des parametres d'une deuxieme plaquette
  int deux_plaq() {
    for (int i=0; i<6; i++) {
      if (deux_plaq_n[i] == 1) return(1);
    }
    return(0);
  }

  // obtenir coordonnees (cm) des traces 0 (unique amas) 1 (haut signal)
  void coord(Trace* tra) {
    //    printf("coord\n");
    for (int ii=0; ii<6; ii++) {
      int ax = 0;
      int ay = 0;
      if (ii < 4) {
        if (tra->xp[ii] == -999.) tra->x[ii] = tra->xp[ii];
        else {
          tra->x[ii] = (tra->xp[ii]-1.)*0.0110;
          ax++;
        }
        if (tra->yp[ii] == -999.) tra->y[ii] = tra->yp[ii];
        else {
          if (deux_plaq_n[ii] == 0)
            tra->y[ii] = (tra->yp[ii]-ppiste_n[ii])*0.0208;
          else {
            if (tra->yp[ii] < 833.) {
              if (tra->yp[ii] < 642.)
                tra->y[ii] = (tra->yp[ii]-641.)*1.5*0.0208;
              else
                tra->y[ii] = (tra->yp[ii]-642.)*0.0208 + 0.0208*1.5;
                tra->y[ii]-=delta_y_ind4;
              //              printf("yp %f y %f\n",tra->yp[ii],tra->y[ii]);
            }
            else {
              if (tra->yp[ii] <= 1023.) {
                if (tra->yp[ii] < 834.)
                  tra->y[ii] = (tra->yp[ii]-ppiste_n[ii])*1.5*0.0208;
                else
                  tra->y[ii] = (tra->yp[ii]-ppiste_n[ii]+1)*0.0208 + 1.5*0.0208;
                }
              else
```

```
              tra->y[ii] = (1023.-ppiste_n[ii]+1)*0.0208 + 1.5*0.0208
                + (tra->yp[ii]-1023.)*0.0208*1.5;
            }
          }
          ay++;
        }
      } else {
         if (tra->xp[ii] == -999.) tra->x[ii] = tra->xp[ii];
         else {
           tra->x[ii] = (tra->xp[ii]-1.)*0.0050;
           ax++;
         }
         if (tra->yp[ii] == -999.) tra->y[ii] = tra->yp[ii];
         else {
           tra->y[ii] = (tra->yp[ii]-ppiste_n[ii])*0.0050;
           ay++;
         }
      }
      tra->z[ii]=algpar[14][ii];

      // si seulement amas y
      if (ax == 0 && ay == 1) {
         apos(ii,1,tra);
         apos(ii,2,tra);
         break;
      }
      // si seulement amas x
      if (ax == 1 && ay == 0) {
         apos(ii,0,tra);
         apos(ii,2,tra);
         break;
      }

      // correction z pour les angles d'inclinaision en les plans xz et yz
      //      printf(" ii %d dxdz %f sin %f\n",ii,algpar[10][ii],sind(algpar[10]
[ii]));
      float dzdx = (tra->x[ii]-plaq_dim[0]*algpar[11][ii])*sind(algpar[10][ii]);
      float pivoty = algpar[13][ii];
      if (deux_plaq_n[ii] && tra->yp[ii] > 832. ) pivoty = algpar_2[13][ii];
      float dzdy = (tra->y[ii]-plaq_dim[1]*pivoty)*sind(algpar[11][ii]);
      float dz = (dzdx*dzdx) + (dzdy*dzdy);
      if (dz > 0.) dz=sqrt(dz);
      tra->z[ii]-=dz;
      if (ii == 3) depz = dz;
      apos(ii,0,tra);
      apos(ii,1,tra);
      apos(ii,2,tra);
      if (ii!=ref2) apos_dxy_dyx(ii,tra);
      //      printf("i %d x %f y %f z %f\n",ii,tra->x[ii],tra->y[ii],tra->z[ii]
);
    }
  }

  // retourne positions apres alignement i-detecteur(0-3), j-0(x),1(y),2(z)
  // pour x/y apres au niveau des corrections dxx, dyy
  void apos(int i, int j, Trace* tra) {
    // en cm
    if (j==0) {
       tra->x[i]-=algpar[j][i];
       tra->xsa[i] = tra->x[i];
      //       printf("even %d i %d  y % e y0 %e diff %e\n",nevent,i,x[i],algpar
[0][i],val);
       //        if (i==3) val-=(algpar[2][i]*(x[i]-algpar[0][i])+algpar[3][i]);
       //        if (i==3) val-=(algpar[2][i]*(x[i]-algpar[0][i])+algpar[3][i]);
       if (i==ind3 || i==ind4 || i==indp1 || i==indp2) {
          tra->x[i]-=(algpar[2][i]*tra->x[i]+algpar[3][i]);
          xdx[i] = tra->x[i];
       }
       return;
```



```
    }
    if (j==1) {
      if (deux_plaq_n[i] == 0)
         tra->y[i]-=algpar[j][i];
      else {
        if (tra->yp[i] < 833.)
          tra->y[i]-=algpar[j][i];
        else
         tra->y[i]-=algpar_2[j][i];
      }
      tra->ysa[i] = tra->y[i];
      //      printf("i %d  y % e y0 %e diff %e\n",i,y[i],algpar[1][i],val);
       if (i==ind4 || i==indp1 || i==indp2) {
          tra->y[i]-=(algpar[4][i]*tra->y[i]+algpar[5][i]);
          ydy[i] = tra->y[i];
       }
       if (i==ind3) {
         if (deux_plaq_n[i] == 0) {
           tra->y[i]-=(algpar[4][i]*tra->y[i]+algpar[5][i]);
           ydy[i] = tra->y[i];
         }
         else {
           if (tra->yp[i] < 833) {
             tra->y[i]-=(algpar[4][i]*tra->y[i]+algpar[5][i]);
             ydy[i] = tra->y[i];
           }
           else {
             tra->y[i]-=(algpar_2[4][i]*tra->y[i]+algpar_2[5][i]);
             ydy[i] = tra->y[i];
           }
         }
       }

       return;
    }
    return;
  }

  // retourne positons x/y apres les corrections dxy et dyx
  void apos_dxy_dyx(int i, Trace* tra) {
    // en cm
    if (i==ref1) {
      xdx[i] = tra->x[i];
      ydy[i] = tra->y[i];
      tra->x[i]-=(algpar[6][i]*ydy[i]+algpar[7][i]);
      tra->y[i]-=(algpar[8][i]*xdx[i]+algpar[9][i]);
    }
    if (i==ind3 || i==ind4 || i==indp1 || i==indp2) {
      tra->x[i]-=(algpar[6][i]*ydy[i]+algpar[7][i]);
      tra->y[i]-=(algpar[8][i]*xdx[i]+algpar[9][i]);
    }
    return;
  }

  // retourne la position predite pour la trace dans l'intervalle entre les pist
es de lecture
  float ipred(int i, int j, Trace* tra) {
    // en cm
    if (j==0) {
      float val = xpospred[i];
      if (i==ref1) val+=(algpar[6][i]*ydy[i]+algpar[7][i]);
      if (i==ind3 || i==ind4 || i==indp1 || i==indp2) {
        val+=(algpar[6][i]*ydy[i]+algpar[7][i]);
        val+=(algpar[2][i]*val+algpar[3][i]);
      }
      val+=algpar[j][i];
      if (i==indp1  || i==indp2)
         val = (val/0.0050+1.);
      else
```

```
        val = (val/0.0110+1.);
      return(val);
    }
    if (j==1) {
      float val = yospred[i];
      if (i==ref1) val+=(algpar[8][i]*xdx[i]+algpar[9][i]);
      if (i==ind3 || i==ind4 || i==indp1 || i==indp2) {
        val+=(algpar[8][i]*xdx[i]+algpar[9][i]);
        val+=(algpar[4][i]*val+algpar[5][i]);
      }
      val+=algpar[j][i];
      if (i==indp1 || i==indp2)
         val = (val/0.0050+ppiste_n[i]);
      else
         val = (val/0.0208+ppiste_n[i]);
      return(val);
    }
    return(-1.);
  }

  // retourne positons apres alignement i-detecteur(0-3), j-0(x),1(y),2(z)
  double apos(int h, int i, int j, Trace* tra) {
    // en cm
    //    printf("j %d\n",j);
    if (j==0) {
       double val = tra->x[i]-algpar_fich[h][0][i];
       //       printf("even %d i %d  y % e y0 %e diff %e\n",nevent,i,x[i],algpar
_fich[h][0][i],val);
       if (i==3) val-=(algpar_fich[h][2][i]*(tra->x[i]-algpar_fich[h][0][i])+alg
par_fich[h][3][i]);
       if (i==3) val-=(algpar_fich[h][6][i]*(tra->y[i]-algpar_fich[h][1][i])+alg
par_fich[h][7][i]);
       if (h==2 && i==2) val-=(algpar_fich[h][6][i]*(tra->y[i]-algpar_fich[h][1]
[i])+algpar_fich[h][7][i]);
       if (i==0) val-=(algpar_fich[h][6][i]*(tra->y[i]-algpar_fich[h][1][i])+alg
par_fich[h][7][i]);
       return (val);
    }
    if (j==1) {
      //       printf("gy %f\n",val);
      double val = tra->y[i]-algpar_fich[h][1][i];
      //      printf("i %d  y % e y0 %e diff %e\n",i,y[i],algpar_fich[h][1][i],v
al);
       return (val);
    }
    if (j==2) return(algpar[10][i]);
    return(-1.);
  }

  void EcrireAlgPar();

  void LireAlgPar();

  void LireFichAlgPar(int run);

  void LireFichAmasPar(int run);

  void LireFonctionEta(int run);

  void LireFonctionEta3(int run);

  // position x a 'k' du ligne entre 'i' and 'j'
  double ligne_projx(int i, int j, int k, Trace* tra);
  // position y a 'k' du ligne entre 'i' and 'j'
  double ligne_projy(int i, int j, int k, Trace* tra);

  // position x a z du ligne entre 'i' and 'j'
  double ligne_projxz(int i, int j, float z, Trace* tra);
  // position y a z du ligne entre 'i' and 'j'
```



```
  double ligne_projyz(int i, int j, float z, Trace* tra);

  // divergence dx/dz entre positions i,j
  double div_dxdz(int i, int j, Trace* tra);
  // divergence dy/dz entre positions i,j
  double div_dydz(int i, int j, Trace* tra);

  // divergence dx/dz pour toutes les position avec fit
  double div_dxdz_lin(Trace* tra, int exclu);
  // divergence dx/dz avec les reference et une echelle
  void div_dxdz_lin_ref(Trace* tra);
  // divergence dx/dz entre les positions 1-3 avec fit
  double div_dxdz_lin_pos123(Trace* tra, int exclu);
  // residus en x pour le petit a la position 0
  double div_dxdz_lin_proj_pos0(Trace* tra);
  // residus en x pour le petit a la position 4
  double div_dxdz_lin_proj_pos4(Trace* tra);
  // residus en x pour l'efficacite
  int div_dxdz_lin_eff(Trace* tra, int exclu);
  // divergence dy/dz pour toutes les positions avec fit
  double div_dydz_lin(Trace* tra, int exclu);
  // divergence dy/dz avec les reference et une echelle
  void div_dydz_lin_ref(Trace* tra);
  // divergence dy/dz entre positions 1-3 avec fit
  double div_dydz_lin_pos123(Trace* tra, int exclu);
  // residus en y pour le petit a la position 0
  double div_dydz_lin_proj_pos0(Trace* tra);
  // residus en y pour le petit a la position 4
  double div_dydz_lin_proj_pos4(Trace* tra);
  // residus en y pour l'efficacite
  int div_dydz_lin_eff(Trace* tra, int exclu);
  // conversion deg->grad per sin(x)
  float sind(float deg) {
    return(sin(3.14159*deg/180.));}

  ClassDef(Align,1)
};


#endif
```