Appendix A

Uncertainty Calculations

The normalized factorial moments are defined as

$$F_{j,l} = \frac{\langle n_c(n_c-1)\ldots(n_c-j+1)n_g(n_g-1)\ldots(n_g-l+1)\rangle}{\langle n_c\rangle^j \langle n_g\rangle^l} \tag{A.1}$$

where $\langle \mathcal{O} \rangle = \frac{1}{N}\sum_{n_c,n_g} \mathcal{O}\mathcal{N}(n_c,n_g)$, $N = \sum_{n_c,n_g} \mathcal{N}(n_c,n_g)$, and $\mathcal{N}(n_c,n_g)$ is the number of events with n_c charged tracks and n_g gammas.

We assume that $\sigma^2_{\mathcal{N}(n_c,n_g)} = \mathcal{N}(n_c,n_g)$. The uncertainty in $F_{j,l}$ is then given by

$$\sigma^2_{F_{j,l}} = \frac{1}{N}\left[\frac{\langle [n_c\ldots(n_c-j+1)n_g\ldots(n_g-l+1)]^2\rangle}{\langle n_c\rangle^{2j}\langle n_g\rangle^{2l}} - (j+l-1)^2 {F_{j,l}}^2\right.$$

$$+j^2\frac{\langle n_c^2\rangle}{\langle n_c\rangle^2}{F_{j,l}}^2 + l^2\frac{\langle n_g^2\rangle}{\langle n_g\rangle^2}{F_{j,l}}^2 + 2jl\frac{\langle n_c n_g\rangle}{\langle n_c\rangle\langle n_g\rangle}{F_{j,l}}^2$$

$$-2j\frac{\langle n_c[n_c\ldots(n_c-j+1)n_g\ldots(n_g-l+1)]\rangle}{\langle n_c\rangle^{j+1}\langle n_g\rangle^{l}}F_{j,l}$$

$$\left.-2l\frac{\langle n_g[n_c\ldots(n_c-j+1)n_g\ldots(n_g-l+1)]\rangle}{\langle n_c\rangle^{j}\langle n_g\rangle^{l+1}}F_{j,l}\right] \tag{A.2}$$

In order to find the correlated uncertainty for any two F's we define

$$x = F_{j,l} + F_{i,m} \tag{A.3}$$

which has uncertainty given by

$$\sigma_x^2 = \sigma_{F_{j,l}}^2 + \sigma_{F_{i,m}}^2 + 2\sigma_{F_{j,l}F_{i,m}} \tag{A.4}$$

Writing the F's in terms of the $\mathcal{N}(n_c, n_g)$, we get

$$\sigma_x^2 = \sum_{n_c', n_g'}\left[\frac{n_c'\ldots(n_c'-j+1)n_g'\ldots(n_g'-l+1)}{N\langle n_c\rangle^{j}\langle n_g\rangle^{l}} + (j+l-1)\frac{F_{j,l}}{N}\right.$$

$$-j\frac{n_c'F_{j,l}}{N\langle n_c\rangle} - l\frac{n_g'F_{j,l}}{N\langle n_g\rangle}$$

$$+\frac{n_c'\ldots(n_c'-i+1)n_g'\ldots(n_g'-m+1)}{N\langle n_c\rangle^{i}\langle n_g\rangle^{m}} + (i+m-1)\frac{F_{i,m}}{N}$$

$$-i\frac{n_c' F_{i,m}}{N\langle n_c\rangle} - m\frac{n_g' F_{i,m}}{N\langle n_g\rangle}\Bigg]^2 \mathcal{N}(n_c', n_g') \tag{A.5}$$

and we can pick out the correlation terms to find

$$\sigma_{F_{j,l}F_{i,m}} = \frac{1}{N}\Bigg[\frac{\langle [n_c\ldots(n_c-j+1)n_g\ldots(n_g-l+1)][n_c\ldots(n_c-i+1)n_g\ldots(n_g-m+1)]\rangle}{\langle n_c\rangle^{j+i}\langle n_g\rangle^{l+m}}$$

$$-i\frac{\langle n_c[n_c\ldots(n_c-j+1)n_g\ldots(n_g-l+1)]\rangle}{\langle n_c\rangle^{j+1}\langle n_g\rangle^{l}}F_{i,m}$$

$$-m\frac{\langle n_g[n_c\ldots(n_c-j+1)n_g\ldots(n_g-l+1)]\rangle}{\langle n_c\rangle^{j}\langle n_g\rangle^{l+1}}F_{i,m}$$

$$-j\frac{\langle n_c[n_c\ldots(n_c-i+1)n_g\ldots(n_g-m+1)]\rangle}{\langle n_c\rangle^{i+1}\langle n_g\rangle^{m}}F_{j,l}$$

$$-l\frac{\langle n_g[n_c\ldots(n_c-i+1)n_g\ldots(n_g-m+1)]\rangle}{\langle n_c\rangle^{i}\langle n_g\rangle^{m+1}}F_{j,l}$$

$$+ij\frac{\langle n_c^2\rangle}{\langle n_c\rangle^2}F_{j,l}F_{i,m} + lm\frac{\left\langle n_g^2\right\rangle}{\langle n_g\rangle^2}F_{j,l}F_{i,m}$$

$$+(jm+il)\frac{\langle n_c n_g\rangle}{\langle n_c\rangle\langle n_g\rangle}F_{j,l}F_{i,m}-(j+l-1)(i+m-1)F_{j,l}F_{i,m}\Bigg] \quad \text{(A.6)}$$

The generalized r defined by

$$r_{j,l}=\frac{F_{j,l}}{F_{j+l,0}} \quad \text{(A.7)}$$

has uncertainty given by

$$\sigma^2_{r_{j,l}}=\left(\frac{1}{F_{j+l,0}}\right)^2\sigma^2_{F_{j,l}}+\left(-\frac{r_{jl}}{F_{j+l,0}}\right)^2\sigma^2_{F_{j+l,0}}+2\left(-\frac{r_{jl}}{F^2_{j+l,0}}\right)\sigma_{F_{j,l}F_{j+l,0}} \quad \text{(A.8)}$$

Appendix B

Track fitting code

B.1 Track fitter: fit.f

```
      SUBROUTINE fit_init

      IMPLICIT NONE

      INTEGER I,NN
      REAL sigsq,wiresp,pi,uang,sum,cu,su
      REAL U0(24),V0(24),ang(24),Z(24)

      common/cham_param/NN,sigsq,wiresp,pi,uang,sum,cu,su,U0,V0,ang,Z

C  position of the center of wire 0 and angle of rotation in (x,y,z)
      real zz(24)/123.69,126.57,129.69,132.57,
     +         135.69,138.69,141.69,144.69,
     +         167.09,169.89,172.84,176.09,
     +         178.84,181.84,185.09,189.65,
     +         194.29,197.29,200.29,203.29,
     +         206.16,209.16,212.41,215.41/
      real xx0(24)/2.806,10.117,0.853,11.792,
     +         9.997,0.818,10.076,1.252,
     +         1.692,10.090,10.970,1.535,
     +         11.199,12.360,11.424,3.366,
     +         11.602,11.327,2.814,12.648,
     +         12.056,3.560,3.140,2.681/
      real yy0(24)/11.653,11.032,10.870,9.255,
     +         10.910,9.193,11.700,1.813,
     +         2.568,12.305,11.702,3.508,
```

```
     +          11.781,10.771,11.812,1.695,
     +          12.036,12.524,2.963,11.415,
     +          12.114,1.613,3.048,3.748/
        real aangle(24)/-1.0315,-2.3422,-0.7575,-2.5953,
     +          -2.3318,-0.5044,-2.0857,0.8029,
     +          0.7959,-2.1084,-2.3370,0.5794,
     +          -2.3370,-2.5063,-2.3405,0.9669,
     +          -2.3370,-2.2201,0.8029,-2.4574,
     +          -2.3527,1.0420,0.7837,0.6266/


        sigsq=1./12.  !uncer in location of track (wire spacings) for each hit
        wiresp=0.1    !wire spacing in inches
        NN=4
        pi=3.141592654
        uang=45.43*pi/180.  !taken as mean angle of "u chambers"
        cu=cos(uang)
        su=sin(uang)

        do i=1,24
        U0(i)=xx0(i)*cu+yy0(i)*su    !u,v of center of wire 0
        V0(i)=-xx0(i)*su+yy0(i)*cu
        Z(i)=zz(i)
        if (aangle(i).lt.2*pi) aangle(i)=aangle(i)+2*pi
        ang(i)=aangle(i)-uang
        ang(i)=ang(i)-pi/2.
        enddo

        RETURN
        END

c----------------------------------------------------------------

        SUBROUTINE fit_wires

        IMPLICIT NONE

        INTEGER I,J,K,L,num
        INTEGER icham,ichlast,ncham,iwire,nwires(24),iicham(24)
        REAL wire
        REAL awires(24,128)
        REAL*8 X(4,4),XX(100,4),Y(4),YY(100),a(4),XINV(4,4)
        REAL*8 INDEX(4),D
```

```
      INTEGER NN
      REAL sigsq,wiresp,pi,uang,sum,cu,su
      REAL U0(24),V0(24),ang(24),Z(24)
      common/cham_param/NN,sigsq,wiresp,pi,uang,sum,cu,su,U0,V0,ang,Z

      INTEGER HLIM,HBSIZE
      PARAMETER(HBSIZE=500000)
      COMMON/PAWC/HLIM(HBSIZE)

      include 'wires.inc'
      include 'fit.inc'

      include 'track_dat.inc'
      include 'dst.inc'

      integer ib(24),dmin,newwire
      include 'EVENT.INC'    !to find missing hit in front-8 tracks

      DO I=1,ntrack
100   continue

C  will fit track parameters 'a' in matrix eq Xa=Y

      DO J=1,4
        DO K=1,4
        X(J,K)=0
        ENDDO
        Y(J)=0
      ENDDO

C  will need to know which chambers and which wires in each chamber
C  were hit to get chisq
      ncham=0
      ichlast=0
      if (nnwire(i).gt.100) write(6,*)'nnwire=',nnwire(i)
      DO J=1,nnwire(i)                      !go through hit wires from dst
          icham=int((wires(I,J)-1)/128.)+1
          iwire=wires(I,J)-(icham-1)*128
        IF (icham.ne.ichlast) THEN
          ncham=ncham+1
          iicham(ncham)=icham
          nwires(ncham)=1
        ELSE
          nwires(ncham)=nwires(ncham)+1
```

```
          ENDIF
          awires(ncham,nwires(ncham))=iwire
          ichlast=icham

C  fit XX(i,j)a(j)=YY(i) for all chambers i to get a(j)
C  min chisq -> Y=Xa, a=(Xinv)Y
C  (take same uncertainty for all hits siqsq=1/12 (wiresp^2))

          XX(J,1)=-sin(ang(icham))
          XX(J,2)=-Z(icham)*sin(ang(icham))
          XX(J,3)=cos(ang(icham))
          XX(J,4)=Z(icham)*cos(ang(icham))
          YY(J)=-U0(icham)*sin(ang(icham))+V0(icham)*cos(ang(icham))+
     +          iwire*wiresp
          DO L=1,4
            DO K=1,4
              X(L,K)=X(L,K)+XX(J,L)*XX(J,K)
            ENDDO
            Y(L)=Y(L)+XX(J,L)*YY(J)
          ENDDO
999     continue
        ENDDO

C  invert X
        DO J=1,4
          DO K=1,4
            XINV(J,K)=0
          ENDDO
          XINV(J,J)=1
        ENDDO
C  Numerical Recipes subroutines to invert a matrix
        CALL LUDCMP(X,NN,INDEX,D)
        DO K=1,4
          CALL LUBKSB(X,NN,INDEX,XINV(1,K))
        ENDDO

        DO J=1,4
          a(J)=0
          DO K=1,4
            a(J)=a(J)+Y(K)*XINV(J,K)    !get track parameters
          ENDDO
        ENDDO

C  fill fit.inc
```

```
      a_u(I)=a(1)
      b_u(I)=a(2)
      a_v(I)=a(3)
      b_v(I)=a(4)


C  get chisq (if more than one wire in a chamber is part of the track,
C   use residuals from each wire
C   total number of wires hit = num)
      chisq(I)=0
      num=0
      DO J=1,ncham
        K=iicham(J)
        wire=(-(a_u(I)+b_u(I)*Z(K)-U0(K))*sin(ang(K))
     +     +(a_v(I)+b_v(I)*Z(K)-V0(K))*cos(ang(K)))/wiresp
        DO L=1,nwires(J)
          num=num+1
          chisq(I)=chisq(I)+(wire-awires(J,L))**2
        ENDDO
      ENDDO
      chisq(I)=chisq(I)/(sigsq*(num-4))

C  get covariance matrix
      DO J=1,4
      DO K=1,4
        covar(I,J,K)=(sigsq*0.01)*XINV(J,K)
      ENDDO
      ENDDO

  enddo
  endif

      ENDDO

      RETURN
      END

C  ----------------------------------------------------

     SUBROUTINE LUDCMP(A,N1,INDX,D)    !LU decomposition
      IMPLICIT NONE
      integer n,i,j,k,imax,N1
      real*8 nmax,tiny,aamax,sum,dum !need double precision to get XinvX=1
```

```
      PARAMETER (NMAX=46,TINY=1.0E-20)
      REAL*8 NP,D,A(4,4),INDX(4),VV(4)

      N=N1
      D=1.
      DO 12 I=1,N
      AAMAX=0.
      DO 11 J=1,N
        IF (ABS(A(I,J)).GT.AAMAX) AAMAX=ABS(A(I,J))
11    CONTINUE
      IF (AAMAX.EQ.0.) PAUSE 'Singular matrix.'
      VV(I)=1./AAMAX
12    CONTINUE
      DO 19 J=1,N
      IF (J.GT.1) THEN
        DO 14 I=1,J-1
          SUM=A(I,J)
          IF (I.GT.1)THEN
            DO 13 K=1,I-1
              SUM=SUM-A(I,K)*A(K,J)
13          CONTINUE
            A(I,J)=SUM
          ENDIF
14      CONTINUE
      ENDIF
      AAMAX=0.
      DO 16 I=J,N
        SUM=A(I,J)
        IF (J.GT.1)THEN
          DO 15 K=1,J-1
            SUM=SUM-A(I,K)*A(K,J)
15        CONTINUE
          A(I,J)=SUM
        ENDIF
        DUM=VV(I)*ABS(SUM)
        IF (DUM.GE.AAMAX) THEN
          IMAX=I
          AAMAX=DUM
        ENDIF
16    CONTINUE
      IF (J.NE.IMAX)THEN
        DO 17 K=1,N
          DUM=A(IMAX,K)
          A(IMAX,K)=A(J,K)
```

```
          A(J,K)=DUM
17      CONTINUE
        D=-D
        VV(IMAX)=VV(J)
      ENDIF
      INDX(J)=IMAX
        IF(A(J,J).EQ.0.)A(J,J)=TINY
      IF(J.NE.N)THEN
        DUM=1./A(J,J)
        DO 18 I=J+1,N
          A(I,J)=A(I,J)*DUM
18      CONTINUE
      ENDIF
19    CONTINUE
      RETURN
      END



    SUBROUTINE LUBKSB(A,N1,INDX,B)     !back substitution
      IMPLICIT NONE
      integer n,ii,i,ll,j,N1
      REAL*8 sum,NP,A(4,4),INDX(4),B(4)
      II=0
      N=N1
      DO 12 I=1,N
      LL=INDX(I)
      SUM=B(LL)
      B(LL)=B(I)
      IF (II.NE.0)THEN
        DO 11 J=II,I-1
          SUM=SUM-A(I,J)*B(J)
11      CONTINUE
      ELSE IF (SUM.NE.0.) THEN
        II=I
      ENDIF
      B(I)=SUM
12    CONTINUE
      DO 14 I=N,1,-1
      SUM=B(I)
      IF(I.LT.N)THEN
        DO 13 J=I+1,N
          SUM=SUM-A(I,J)*B(J)
13      CONTINUE
```

```
        ENDIF
        B(I)=SUM/A(I,I)
14      CONTINUE
        RETURN
        END


C------------------------------------------------------
C  wires.inc  (filled by eg vertexer.f)
        integer ntrack,nnwire(200),wires(200,100)
        common/wiresforfit/ntrack,nnwire,wires

C  (must be filled by user unless using the vertexer)
C  ntrack=# tracks
C  nnwire(i)=# wires in track i
C  wires(i,j)=jth wire of ith track in usual format (icham-1)*128+iwire


C------------------------------------------------------
C  fit.inc (filled by fit.f)
        REAL a_u(200),b_u(200),a_v(200),b_v(200),chisq(200)
        REAL COVAR(200,4,4)
        integer current_evnum
        common/track_param/a_u,b_u,a_v,b_v,chisq,covar,current_evnum

C(determined by fit.f)
C  u=a_u+b_u*z, v=a_v+b_v*z
C
C  for covariance matrix, indices go like
C  1 = a_u, 2 = b_u, 3 = a_v, 4 = b_v
```

B.2 Code for correcting tracks with double hits in the non-u chambers: refit.f

```
      SUBROUTINE refit

      IMPLICIT NONE

      INTEGER I,J,K,L,M
      INTEGER icham,ichlast,ncham,iwire,nwires(24),iicham(24)

      INTEGER NN
      REAL sigsq,wiresp,pi,uang,sum,cu,su
      REAL U0(24),V0(24),ang(24),Z(24)
      common/cham_param/NN,sigsq,wiresp,pi,uang,sum,cu,su,U0,V0,ang,Z

      integer itrack
      real av,bv,v(24),u(24),vlead,ucham(24)
      real sz,szz,sv,szv,cth(24),sth(24),denom,predwire,wire(18000,24)
      real sigth,thgam,thtail
      integer nwirepl(24),iwirepl(24,10),nwire,icomb

      integer iw(8),i1,i2,i3,i4,i5,i6,i7,i8
      integer ncomb,ichi,ipoint
      real chi(18000),point(18000),chimin,pointmin

      integer ntr,hit2
      integer nch1,nch2
      common/newnch/nch1,nch2

      include 'wires.inc'
      include 'fit.inc'

      include 'track_dat.inc'
      include 'dst.inc'

C  get u chambers
      do j=1,24
      ucham(j)=0
      enddo
      ucham(2)=1
      ucham(5)=1
      ucham(8)=1
      ucham(9)=1
```

```
      ucham(11)=1
      ucham(13)=1
      ucham(15)=1
      ucham(17)=1
      ucham(19)=1
      ucham(23)=1
      ucham(24)=1

C  will need orientation of non-u chambers to fit in v
      do k=1,24
        if (ucham(k).eq.0) then
          sth(k)=sin(ang(k))
          cth(k)=cos(ang(k))
        endif
      enddo

C  check number of wires hit per plane
C  if 2 wires hit are separated by more than 3 wires in any plane, refit
      do i=1,ntrack
        do k=1,24
          nwirepl(k)=0
        enddo
        do j=1,nnwire(i)
          icham=int((wires(i,j)-1.)/128.)+1
          nwirepl(icham)=nwirepl(icham)+1
          iwirepl(icham,nwirepl(icham))=wires(i,j)
        enddo
        hit2=0
        do k=1,24
          if ((nwirepl(k).eq.2).and.
     +          (abs(iwirepl(k,1)-iwirepl(k,2)).gt.3)) then
            hit2=1
          endif
        enddo
        if (hit2.eq.0) goto 999

C  how many chambers have "double hits" and which wires are involved
      ncham=0
      do k=1,24
        nwirepl(k)=0
      enddo
      ichlast=0
      DO J=1,nnwire(i)
        icham=int((wires(I,J)-1)/128.)+1
```

```
          if (ucham(icham).eq.0) then
            if (icham.ne.ichlast) ncham=ncham+1
            iicham(ncham)=icham
            nwirepl(ncham)=nwirepl(ncham)+1
            iwirepl(ncham,nwirepl(ncham))=wires(I,J)-(icham-1)*128
            ichlast=icham
          endif
        ENDDO

C  cycle thru all wires!!!
C  and find all possible combinations with 1 wire per plane
        ncomb=0
        if (ncham.eq.3) then
        do i1=1,nwirepl(1)
          iw(1)=i1
        do i2=1,nwirepl(2)
          iw(2)=i2
        do i3=1,nwirepl(3)
          iw(3)=i3
          ncomb=ncomb+1
          do j=1,ncham
          if (ncomb.le.18000) wire(ncomb,j)=iwirepl(j,iw(j))
          enddo
        enddo
        enddo
        enddo
        endif
        if (ncham.eq.4) then
        do i1=1,nwirepl(1)
          iw(1)=i1
        do i2=1,nwirepl(2)
          iw(2)=i2
        do i3=1,nwirepl(3)
          iw(3)=i3
        do i4=1,nwirepl(4)
          iw(4)=i4
          ncomb=ncomb+1
          do j=1,ncham
          if (ncomb.le.18000) wire(ncomb,j)=iwirepl(j,iw(j))
          enddo
        enddo
        enddo
        enddo
        enddo
```

```
endif
if (ncham.eq.5) then
do i1=1,nwirepl(1)
  iw(1)=i1
do i2=1,nwirepl(2)
  iw(2)=i2
do i3=1,nwirepl(3)
  iw(3)=i3
do i4=1,nwirepl(4)
  iw(4)=i4
do i5=1,nwirepl(5)
  iw(5)=i5
  ncomb=ncomb+1
  do j=1,ncham
  if (ncomb.le.18000) wire(ncomb,j)=iwirepl(j,iw(j))
  enddo
enddo
enddo
enddo
enddo
enddo
endif
if (ncham.eq.6) then
do i1=1,nwirepl(1)
  iw(1)=i1
do i2=1,nwirepl(2)
  iw(2)=i2
do i3=1,nwirepl(3)
  iw(3)=i3
do i4=1,nwirepl(4)
  iw(4)=i4
do i5=1,nwirepl(5)
  iw(5)=i5
do i6=1,nwirepl(6)
  iw(6)=i6
  ncomb=ncomb+1
  do j=1,ncham
  if (ncomb.le.18000) wire(ncomb,j)=iwirepl(j,iw(j))
  enddo
enddo
enddo
enddo
enddo
enddo
```

```
enddo
endif
if (ncham.eq.7) then
do i1=1,nwirepl(1)
  iw(1)=i1
do i2=1,nwirepl(2)
  iw(2)=i2
do i3=1,nwirepl(3)
  iw(3)=i3
do i4=1,nwirepl(4)
  iw(4)=i4
do i5=1,nwirepl(5)
  iw(5)=i5
do i6=1,nwirepl(6)
  iw(6)=i6
do i7=1,nwirepl(7)
  iw(7)=i7
  ncomb=ncomb+1
  do j=1,ncham
  if (ncomb.le.18000) wire(ncomb,j)=iwirepl(j,iw(j))
  enddo
enddo
enddo
enddo
enddo
enddo
enddo
enddo
endif
if (ncham.eq.8) then
do i1=1,nwirepl(1)
  iw(1)=i1
do i2=1,nwirepl(2)
  iw(2)=i2
do i3=1,nwirepl(3)
  iw(3)=i3
do i4=1,nwirepl(4)
  iw(4)=i4
do i5=1,nwirepl(5)
  iw(5)=i5
do i6=1,nwirepl(6)
  iw(6)=i6
do i7=1,nwirepl(7)
  iw(7)=i7
```

```
      do i8=1,nwirepl(8)
        iw(8)=i8
        ncomb=ncomb+1
        do j=1,ncham
        if (ncomb.le.18000) wire(ncomb,j)=iwirepl(j,iw(j))
        enddo
      enddo
      enddo
      enddo
      enddo
      enddo
      enddo
      enddo
      enddo
      endif
      if (ncomb.gt.18000)  then     !if too many combinations, skip it
        write(6,*)'ncomb=',ncomb
        return
      endif

C  get chisq and pointing in v for each combination
      chimin=99999.
      pointmin=9999.
      DO M=1,ncomb
      Szz=0
      Sz=0
      Sv=0
      Szv=0
      DO j=1,ncham
        k=iicham(j)
        u(k)=a_u(i)+b_u(i)*Z(k)
        v(k)=(u(k)*sth(k)-U0(k)*sth(k)+V0(k)*cth(k)+
     + wire(M,j)*wiresp)/cth(k)
        Szz=Szz+Z(k)**2
        Sz=Sz+Z(k)
        Sv=Sv+v(k)
        Szv=Szv+Z(k)*v(k)
      ENDDO
      denom=ncham*Szz-Sz**2
      av=(Szz*Sv-Sz*Szv)/denom
      bv=(ncham*Szv-Sz*Sv)/denom
      chi(M)=0
      DO J=1,ncham
        K=iicham(J)
```

```
        predwire=(-(u(k)-U0(K))*sin(ang(K))
     +          +(av+bv*Z(K)-V0(K))*cth(K))/wiresp
          chi(M)=chi(M)+(predwire-wire(M,J))**2
        ENDDO
        chi(M)=chi(M)/(sigsq*(ncham-2))
        if (chi(M).lt.chimin) then
          chimin=chi(M)
          ichi=M
        endif

        thtail=atan(bv)
        vlead=av+bv*149.8
        thgam=atan(vlead/(149.8+a_u(i)/b_u(i)))
        sigth=sqrt((0.01*sigsq)*ncham/(ncham*Szz-Sz**2))*
     + cos(thtail)**2
        point(M)=abs(thgam-thtail)/sigth
        if (point(M).lt.pointmin) then
          pointmin=point(M)
          ipoint=M
        endif
        ENDDO

C  require vpoint < 8 and chisq < 7, give preference to best pointing
        if (ncomb.gt.0) then
        icomb=0
        if ((pointmin.lt.8.).and.(chi(ipoint).lt.7)) then
          icomb=ipoint
        else
          if ((chimin.lt.7).and.(point(ichi).lt.8)) then
          icomb=ichi
          endif
        endif
        nwire=nnwire(i)
        nnwire(i)=0
        if (icomb.gt.0) then      !no good track --> nnwire(i)=0
        ichlast=0
        DO J=1,nwire                            !refill list of wires corresp
          icham=int((wires(I,J)-1)/128.)+1    ! to track
          if (ucham(icham).eq.1) then
            nnwire(i)=nnwire(i)+1
            wires(i,nnwire(i))=wires(I,J)
          else
            if (icham.ne.ichlast) then
            do k=1,ncham
```

```
          if (iicham(k).eq.icham) then
            nnwire(i)=nnwire(i)+1
            wires(i,nnwire(i))=(icham-1)*128+wire(icomb,k)
          endif
        enddo
        endif
      endif
      ichlast=icham
    ENDDO
    endif
    endif


999     continue
        enddo

C  refill wires.inc
        ntr=ntrack
        ntrack=0
        nch1=0
        do i=1,ntr
          if (nnwire(i).gt.0) then
            ntrack=ntrack+1
            if (i.le.nch) nch1=nch1+1
            nnwire(ntrack)=nnwire(i)
            do j=1,nnwire(i)
              wires(ntrack,j)=wires(i,j)
            enddo
          endif
        enddo

C  refit tracks from new wires.inc (for all tracks is cpu intensive but easier)
        call fit_wires


        RETURN
        END
```

Appendix C

Vertexer code

C.1 Vertexer: vertexer.f

```
SUBROUTINE vertexer

IMPLICIT NONE

INTEGER HLIM,HBSIZE
PARAMETER(HBSIZE=500000)
COMMON/PAWC/HLIM(HBSIZE)

integer i,j,k,l,m,keep

real M1(50,2,2),M1INV(50,2,2),MM(2,2),MMINV(2,2)
real dchisq,umean,vmean,denom

integer nmatch,imatch(50),ismatched(100)
integer nnclust(50),iiclust(50,50),iclust,new,newtrack,nmax,nncl
integer dstnumb(100),nv

integer max,imax,nsame,nsameu,maxu,icham,ichlast
integer nback(100),nb(24)
real thtail,sigth,thgam,zlead,sigu0

integer ntr,nhead,ntail
real upoint,vpoint,upoint2,vpoint2

integer nch1,nch2
common/newnch/nch1,nch2
```

```
      real uold1,vold1
      real dumin,dvmin,dmin,du,dv,uold,vold
      common/taildist/dumin,dvmin,dmin,du,dv,uold,vold

      include 'fit.inc'
      include 'wires.inc'
      include 'vrtx.inc'
      include 'VRTCS.INC'

      include 'track_dat.inc'
      include 'dst.inc'

      zlead=149.82

C  fill wires.inc with all track segments
      ntrack=0
      DO k=1,nch+nnu
C  get number of wires (don't distinguish betwen heads or tails)
      IF (k.le.nch) THEN
        ntrack=ntrack+1
      if (ntrack.gt.100) write(6,*)ntrack
        dstnumb(ntrack)=k
        nnwire(k)=int(charged(k,6))
        DO J=1,nnwire(ntrack)
          wires(k,j)=wires_c(k,J)
        ENDDO
      ELSE
        DO L=9,24
          nb(L)=0
        ENDDO
        DO j=1,neutral(k-nch,6)
          icham=int((wires_n(k-nch,j)-1)/128.)+1
          nb(icham)=1
        ENDDO
        nback(ntrack+1)=0
        DO L=9,24
          nback(ntrack+1)=nback(ntrack+1)+nb(L)
        ENDDO
        if (nback(ntrack+1).ge.8) then  !can change to exclude mid-8 tracks
          ntrack=ntrack+1
          dstnumb(ntrack)=k
          nnwire(ntrack)=int(neutral(k-nch,6))
          DO J=1,nnwire(ntrack)
```

```
              wires(ntrack,j)=wires_n(k-nch,J)
      ENDDO
          endif
        ENDIF
        ENDDO

C  fit tracks
        call fit_wires
        call refit

C  get position of tracks at lead and uncertainty in pos
        call vrtx(zlead)

C  remove heads which don't point and tracks which don't hit lead
        ntr=ntrack
        ntrack=0
        nch2=0
        do i=1,ntr
          if (i.le.nch1) then
            upoint=-a_u(i)/b_u(i)
          else
            upoint=0
          endif
C  for now, don't demand tails point, loose on heads
          if ((upoint.gt.-50).and.(upoint.lt.60.)) then
            if (i.le.nch1) then
              thtail=atan(b_v(i))
              thgam=atan(v(i)/(zlead+a_u(i)/b_u(i)))
              sigth=sqrt(covar(i,4,4))*cos(thtail)**2
              vpoint=abs(thgam-thtail)/sigth
            else
              vpoint=0
            endif
          if ((vpoint.lt.8.).and.(chisq(i).lt.20.).and.
     +         (u(i).gt.3.3).and.(u(i).lt.11.3).and.
     +          (v(i).gt.-4.0).and.(v(i).lt.4.0)) then
C  refill wires.inc with kept track segments
            ntrack=ntrack+1
            dstnumb(ntrack)=dstnumb(i)
            if (i.le.nch1) nch2=nch2+1  !number of heads which point
            nnwire(ntrack)=nnwire(i)
            do j=1,nnwire(i)
              wires(ntrack,j)=wires(i,j)
            enddo
```

```
            a_u(ntrack)=a_u(i)
            b_u(ntrack)=b_u(i)
            a_v(ntrack)=a_v(i)
            b_v(ntrack)=b_v(i)
            do j=1,4
            do k=1,4
            covar(ntrack,j,k)=covar(i,j,k)
            enddo
            enddo
            chisq(ntrack)=chisq(i)
            u(ntrack)=u(i)
            v(ntrack)=v(i)
            siguu(ntrack)=siguu(i)
            siguv(ntrack)=siguv(i)
            sigvv(ntrack)=sigvv(i)
          endif
          endif
        enddo


C-------------------------------------------------
C  get i,j for tracks to match
        nvrtcs=0

C increase uncer in pos of tails to account for multiple scattering
        do i=1,ntrack
          ismatched(i)=0
          if (i.gt.nch2) then
            if (siguu(i).lt.0.007) siguu(i)=.007
            if (sigvv(i).lt.0.092) sigvv(i)=.092
          endif
        enddo

C  first match heads to other segments (smaller uncertainty)
        do i=1,ntrack
          if (i.gt.nch2) goto 555
          nmatch=0
        do j=1,ntrack
          if (j.eq.i) goto 557
          CALL match(i,j,dchisq)
          if ((dchisq/2.).lt.5.)  then
            nmatch=nmatch+1  !number of matches to segment i
            imatch(nmatch)=j
            ismatched(j)=1
```

```
          endif
557     continue
        enddo !do j
          imatch(nmatch+1)=i  !(include track i in list of matched tracks)


C  if this head has already been found to match other segments,
C   combine all matches
        new=1
        do k=1,nvrtcs
        do l=1,nnclust(k)
        do j=1,nmatch+1
          if (imatch(j).eq.iiclust(k,l)) then
            new=0
            iclust=k
          endif
        enddo
        enddo
        enddo
        if (new.eq.1) then
          nvrtcs=nvrtcs+1
          iclust=nvrtcs
          nnclust(iclust)=0
        endif
        nmax=nnclust(iclust)
        do j=1,nmatch+1
        newtrack=1
        do l=1,nmax
          if (imatch(j).eq.iiclust(iclust,l)) newtrack=0
        enddo
        if (newtrack.eq.1) then
          nnclust(iclust)=nnclust(iclust)+1
          iiclust(iclust,nnclust(iclust))=imatch(j)
        endif
        enddo

555     continue
        enddo !do i


C  now match tail-tail to get gammas
        uold1=999.
        do i=1,ntrack
          if ((i.le.nch2).or.(ismatched(i).eq.1)) goto 556
```

```
  nmatch=0
  uold1=u(i)
  vold1=v(i)
  dmin=999.
  dumin=999.
  dvmin=999.
do j=1,ntrack
  if ((j.gt.nch2).and.(j.ne.i).and.(ismatched(j).ne.1)) then
  CALL match(i,j,dchisq)
  if ((dchisq/2.).lt.5.) then
    nmatch=nmatch+1
    imatch(nmatch)=j
  endif
  endif ! j ne i
enddo !do j
  imatch(nmatch+1)=i

new=1
do k=1,nvrtcs
do l=1,nnclust(k)
do j=1,nmatch+1
  if (imatch(j).eq.iiclust(k,l)) then
    new=0
    iclust=k
  endif
enddo
enddo
enddo
if (new.eq.1) then
  nvrtcs=nvrtcs+1
  iclust=nvrtcs
  nnclust(iclust)=0
endif
nmax=nnclust(iclust)
do j=1,nmatch+1
newtrack=1
do l=1,nmax
  if (imatch(j).eq.iiclust(iclust,l)) newtrack=0
enddo
if (newtrack.eq.1) then
  nnclust(iclust)=nnclust(iclust)+1
  iiclust(iclust,nnclust(iclust))=imatch(j)
endif
enddo
```

```
556     continue
        enddo !do i

        if (uold1.lt.11.25) then
          uold=uold1
          vold=vold1
        endif


C  fill VRTCS.INC with found vertices
        nv=0
        do k=1,nvrtcs
          nverts(k,1)=0
          nverts(k,2)=0
          do j=1,nnclust(k)
            i=iiclust(k,j)
            if (i.gt.nch2) then
              nverts(k,2)=nverts(k,2)+1
            else
              nverts(k,1)=nverts(k,1)+1
            endif
          enddo
          if ((nverts(k,1).eq.1).and.(nverts(k,2).gt.0)) goto 123
                           !if (i,n>0) charged track, pointing is already ok
          nncl=nnclust(k)
          nnclust(k)=0
          nhead=nverts(k,1)
          ntail=nverts(k,2)
          nverts(k,2)=0
          nverts(k,1)=0
          do j=1,nncl
            i=iiclust(k,j)
            upoint=-a_u(i)/b_u(i)
            thtail=atan(b_v(i))
            thgam=atan(v(i)/(zlead+a_u(i)/b_u(i)))
            sigth=sqrt(covar(i,4,4))*cos(thtail)**2
            vpoint=abs(thgam-thtail)/sigth
            if (((upoint.gt.-40).and.(upoint.lt.50).and.
     +          (vpoint.lt.8)).or.
     + ((nhead.eq.0).and.(ntail.gt.1).and.
     + (upoint.lt.50))) then
              keep=1
              if ((nhead.eq.0).and.(ntail.eq.1)) then  !tighter on (0,1)'s
```

```
        keep=0
        do l=9,24
          nb(l)=0
        enddo
        do l=1,nnwire(i)
          icham=int((wires(i,l)-1)/128.)+1
          nb(icham)=1
        enddo
        icham=0
        do l=9,24
          icham=icham+nb(l)
        enddo
        if (icham.ge.14) keep=1
             endif

             if (keep.eq.1) then
             nnclust(k)=nnclust(k)+1
             iiclust(k,nnclust(k))=i
             if (i.gt.nch2) then
               nverts(k,2)=nverts(k,2)+1
             else
               nverts(k,1)=nverts(k,1)+1
             endif
             endif
           endif
         enddo
123      continue
         if (nnclust(k).gt.0) then
           nv=nv+1
           nverts(nv,1)=nverts(k,1)
           nverts(nv,2)=nverts(k,2)
           vrtxpos(nv,1)=0
           vrtxpos(nv,2)=0
        if (nnclust(k).gt.20) write(6,*)'nnclust=',nnclust(k)
C  get mean position of vertex
        MM(1,1)=0
        MM(1,2)=0
        MM(2,2)=0
            do j=1,nnclust(k)
              i=iiclust(k,j)
              nverts(nv,j+2)=dstnumb(i)
              nverts(nv,j+52)=i
         M1(j,1,1)=siguu(i)
         M1(j,2,2)=sigvv(i)
```

```
         M1(j,1,2)=siguv(i)
         M1(j,2,1)=M1(j,1,2)
         denom=M1(j,1,1)*M1(j,2,2)-M1(j,1,2)*M1(j,2,1)
         M1INV(j,1,1)=M1(j,2,2)/denom
         M1INV(j,2,2)=M1(j,1,1)/denom
         M1INV(j,1,2)=-M1(j,1,2)/denom
         M1INV(j,2,1)=-M1(j,2,1)/denom
         MM(1,1)=MM(1,1)+M1INV(j,1,1)
         MM(1,2)=MM(1,2)+M1INV(j,1,2)
         MM(2,2)=MM(2,2)+M1INV(j,2,2)
            enddo
            MM(2,1)=MM(1,2)
            denom=MM(1,1)*MM(2,2)-MM(1,2)*MM(2,1)
            MMINV(1,1)=MM(2,2)/denom
            MMINV(2,2)=MM(1,1)/denom
            MMINV(1,2)=-MM(1,2)/denom
            MMINV(2,1)=-MM(2,1)/denom
            umean=0
            vmean=0
            do j=1,nnclust(k)
              i=iiclust(k,j)
              umean=umean+MMINV(1,1)*M1INV(j,1,1)*u(i)+
     +          MMINV(1,2)*M1INV(j,2,1)*u(i)+
     +          MMINV(1,1)*M1INV(j,1,2)*v(i)+
     +          MMINV(1,2)*M1INV(j,2,2)*v(i)
              vmean=vmean+MMINV(2,1)*M1INV(j,1,1)*u(i)+
     +          MMINV(2,2)*M1INV(j,2,1)*u(i)+
     +          MMINV(2,1)*M1INV(j,1,2)*v(i)+
     +          MMINV(2,2)*M1INV(j,2,2)*v(i)
            enddo
            vrtxpos(nv,1)=umean
            vrtxpos(nv,2)=vmean
          endif
        enddo
        nvrtcs=nv

        RETURN
        END

C-----------------------------------------------------------
C get u,v,uncer at lead

        SUBROUTINE vrtx(zlead)
```

```
      include 'fit.inc'
      include 'wires.inc'
      include 'vrtx.inc'

      include 'track_dat.inc'
      include 'dst.inc'


      thu=45.43*3.141592654/180.
      cu=cos(thu)
      su=sin(thu)

      do i=1,ntrack
      u(i)=a_u(i)+b_u(i)*zlead
      v(i)=a_v(i)+b_v(i)*zlead
      siguu(i)=covar(i,1,1)+2*zlead*covar(i,1,2)+zlead**2*covar(i,2,2)
      sigvv(i)=covar(i,3,3)+2*zlead*covar(i,3,4)+zlead**2*covar(i,4,4)
      siguv(i)=covar(i,1,3)+zlead*(covar(i,1,4)+covar(i,2,3))+
     +  zlead**2*covar(i,2,4)
      enddo

      RETURN
      END


C------------------------------------------------------------
C  get chisq of match between pair of tracks

      SUBROUTINE match(i,j,dchisq)

      IMPLICIT NONE

      include 'fit.inc'
      include 'vrtx.inc'

      integer i,j
      real C1(2,2),C2(2,2),C1INV(2,2),C2INV(2,2),C(2,2),CINV(2,2)
      real dchisq,umean,vmean,denom

      integer nch1,nch2
      common/newnch/nch1,nch2

      real dist
      real dumin,dvmin,dmin,du,dv,uold,vold
```

```
      common/taildist/dumin,dvmin,dmin,du,dv,uold,vold

      real upoint,vpoint,upoint2,vpoint2
      real thtail,sigth,thgam,zlead,sigu0

      INTEGER HLIM,HBSIZE
      PARAMETER(HBSIZE=500000)
      COMMON/PAWC/HLIM(HBSIZE)

      C1(1,1)=siguu(i)
      C1(2,2)=sigvv(i)
      C1(1,2)=siguv(i)
      C1(2,1)=C1(1,2)
      C2(1,1)=siguu(j)
      C2(2,2)=sigvv(j)
      C2(1,2)=siguv(j)
      C2(2,1)=C2(1,2)
C invert
      denom=C1(1,1)*C1(2,2)-C1(1,2)*C1(2,1)
      C1INV(1,1)=C1(2,2)/denom
      C1INV(2,2)=C1(1,1)/denom
      C1INV(1,2)=-C1(1,2)/denom
      C1INV(2,1)=-C1(2,1)/denom
      denom=C2(1,1)*C2(2,2)-C2(1,2)*C2(2,1)
      C2INV(1,1)=C2(2,2)/denom
      C2INV(2,2)=C2(1,1)/denom
      C2INV(1,2)=-C2(1,2)/denom
      C2INV(2,1)=-C2(2,1)/denom
      C(1,1)=C1INV(1,1)+C2INV(1,1)
      C(2,2)=C1INV(2,2)+C2INV(2,2)
      C(1,2)=C1INV(1,2)+C2INV(1,2)
      C(2,1)=C(1,2)
      denom=C(1,1)*C(2,2)-C(1,2)*C(2,1)
      CINV(1,1)=C(2,2)/denom
      CINV(2,2)=C(1,1)/denom
      CINV(1,2)=-C(1,2)/denom
      CINV(2,1)=-C(2,1)/denom
      umean=CINV(1,1)*(C1INV(1,1)*u(i)+C2INV(1,1)*u(j))+
     +       CINV(1,2)*(C1INV(2,1)*u(i)+C2INV(2,1)*u(j))+
     +       CINV(1,1)*(C1INV(1,2)*v(i)+C2INV(1,2)*v(j))+
     +       CINV(1,2)*(C1INV(2,2)*v(i)+C2INV(2,2)*v(j))
      vmean=CINV(2,1)*(C1INV(1,1)*u(i)+C2INV(1,1)*u(j))+
     +       CINV(2,2)*(C1INV(2,1)*u(i)+C2INV(2,1)*u(j))+
     +       CINV(2,1)*(C1INV(1,2)*v(i)+C2INV(1,2)*v(j))+
```

```
     +         CINV(2,2)*(C1INV(2,2)*v(i)+C2INV(2,2)*v(j))
        dchisq=(u(i)-umean)*C1INV(1,1)*(u(i)-umean)
     +      +2*(u(i)-umean)*C1INV(1,2)*(v(i)-vmean)
     +      +(v(i)-vmean)*C1INV(2,2)*(v(i)-vmean)
     +      +(u(j)-umean)*C2INV(1,1)*(u(j)-umean)
     +      +2*(u(j)-umean)*C2INV(1,2)*(v(j)-vmean)
     +      +(v(j)-vmean)*C2INV(2,2)*(v(j)-vmean)

        RETURN
        END


C ------------------------------------------------------------
C  vrtx.inc
        real u(200),v(200),siguu(200),sigvv(200),siguv(200)
        common/lead/u,v,siguu,sigvv,siguv

C  (determined by subroutine vrtx in vertexer.f)
C  position and uncertainty of track at z=zlead

C ------------------------------------------------------------
C       VRTCS.INC
        integer nvrtcs
        integer nverts(100,102)
        real vrtxpos(100,2)
        integer ischarged(100),isgamma(100)

        common/fndvrts/nvrtcs,nverts,vrtxpos,ischarged,isgamma

C  (determined by vertexer.f)
C  nvrtcs=number of vertices at zlead
C  nverts(i,1)=number of heads in vertex i
C  nverts(i,2)=number of tails in vertex i
C  nverts(i,2+j)=dst-track-number of track segment j in vertex i
C  nverts(i,52+j)=fitter-track-number of track segment j in vertex i

C  (determined by chgam.f)
C  ischarged(i)/isgamma(i) =1 if vertex is a charged track/ gamma
C                          =0 otherwise
```

C.2 Code for classifying vertices as charged tracks or photon conversions: chgam.f

```
      SUBROUTINE CHGAM

      IMPLICIT NONE

      include 'EVENT.INC'
      include 'fit.inc'
      include 'wires.inc'
      include 'VRTCS.INC'

      integer nch1,nch2
      common/newnch/nch1,nch2
      INTEGER NN
      REAL sigsq,wiresp,pi,uang,sum,cu,su
      REAL U0(24),V0(24),ang(24),Z(24)
      common/cham_param/NN,sigsq,wiresp,pi,uang,sum,cu,su,U0,V0,ang,Z

      INTEGER I, J, IOS, k, ihit, icham, nb(24), keep,ii,jj,kk,nu,nv
      INTEGER nkeep,ikeep(20),imin
      REAL dist,dmin,r
      real mean,sigmean,upoint,sigupoint,wire
      integer itr,itr0,keepcham,keepwire,keeppoint,npoint

      do j=1,nvrtcs
        ischarged(j)=0
        isgamma(j)=0

C  Cyrus's acceptance
      if ((vrtxpos(j,1).gt.4.25).and.(vrtxpos(j,1).lt.10.25).and.
     +          (vrtxpos(j,2).gt.-3.).and.(vrtxpos(j,2).lt.3.)) then
      r=sqrt((vrtxpos(j,1)-7.25)**2+vrtxpos(j,2)**2)
      if (r.lt.4.) then

C  (1,n>0) charged track
        if ((nverts(j,1).ge.1).and.(nverts(j,2).gt.0)) ischarged(j)=1

C  (1,0)'s have to pass tighter cuts
        if ((nverts(j,1).ge.1).and.(nverts(j,2).eq.0)) then
          keep=0

C  require hit all 8 chambers
```

```
          do i=1,nverts(j,1)
          itr0=nverts(j,52+i)
            keepcham=0
            do k=1,8
              nb(k)=0
            enddo
            do k=1,nnwire(nverts(j,52+i))
       nb(int((wires(nverts(j,52+i),k)-1)/128.)+1)=1
            enddo
            icham=0
            do k=1,8
              if (nb(k).eq.0) then
              wire=nint( (-(a_u(itr0)+b_u(itr0)*Z(K)-U0(K))*sin(ang(K))
     +          +(a_v(itr0)+b_v(itr0)*Z(K)-V0(K))*cos(ang(K)))/wiresp )
              dmin=128
              do jj=1,nhits
                if ((mod(chamb_num(jj),100).eq.k).and.
     +             (abs(wire_num(jj)-wire).lt.dmin))
     +           dmin=abs(wire_num(jj)-wire)
              enddo
              if (dmin.le.1) then
                nb(k)=1
                nnwire(itr0)=nnwire(itr0)+1
                wires(itr0,nnwire(itr0))=wire+(k-1)*128
              endif
              endif
        icham=icham+nb(k)
            enddo
            if (icham.eq.8)keepcham=1

C  don't share wires in all 3 u or all 3 v chambers with other head
            keepwire=1
            do jj=1,nvrtcs
            if (jj.ne.j) then
            do ii=1,nverts(jj,1)+nverts(jj,2)
            if (nverts(jj,52+ii).le.nch2) then
            itr=nverts(jj,52+ii)
            nv=0
            nu=0
            do kk=1,nnwire(itr)
            do k=1,nnwire(itr0)
              if (wires(itr,kk).eq.wires(itr0,k)) then
           icham=int((wires(itr0,k)-1)/128.)+1
         if ((icham.eq.1).or.(icham.eq.3).or.(icham.eq.6))
```

```
     +            nv=nv+1
            if ((icham.eq.2).or.(icham.eq.5).or.(icham.eq.8))
     +             nu=nu+1
                endif
              enddo
              enddo
              if ((nv.eq.3).or.(nu.eq.3)) keepwire=0
              endif
              enddo !ii
              endif
              enddo !jj

C  point to mean z of other charged tracks
              if (i.eq.1) then
                mean=0
                sigmean=0
                npoint=0
                do jj=1,nvrtcs
                if ((nverts(jj,1).ge.1).and.(nverts(jj,2).ge.1)) then
                do ii=1,nverts(jj,1)+nverts(jj,2)
                if (nverts(jj,52+ii).le.nch2) then
                  npoint=npoint+1
                  itr=nverts(jj,52+ii)
                  upoint=-a_u(itr)/b_u(itr)
                  sigupoint=( covar(itr,1,1)+(a_u(itr)/b_u(itr))**2*
     +   covar(itr,2,2)-2*(a_u(itr)/b_u(itr))*covar(itr,1,2) )
     +   /b_u(itr)**2
                mean=mean+upoint/sigupoint
                sigmean=sigmean+1./sigupoint
                endif
                enddo !ii
              endif
              enddo !jj
              if (npoint.gt.0) then
                sigmean=1./sigmean
                mean=mean*sigmean
              endif
              endif !i=1
              keeppoint=1
              if (npoint.gt.0) then
                dist=((-a_u(itr0)/b_u(itr0))-mean)
     +            /sqrt(sigmean+( covar(itr0,1,1)+(a_u(itr0)/
     +            b_u(itr0))**2*covar(itr0,2,2)-2*(a_u(itr0)/
     +            b_u(itr0))*covar(itr0,1,2) )/b_u(itr0)**2 )
```

```
              if (abs(dist).gt.2.) keeppoint=0
          endif !npoint>0
          if ((keepcham.eq.1).and.(keepwire.eq.1).and.
     +       (keeppoint.eq.1)) keep=1
        enddo !i

        if (keep.eq.1) ischarged(j)=1

        endif ! (n,0)

C  (0,n) gamma
        if ((nverts(j,1).eq.0).and.(nverts(j,2).gt.0)) isgamma(j)=1

      endif !vrtxpos
      endif !r<4
      enddo !j


      RETURN
      END
```