August 12, 1987

Internal Note

A Short Guide to MC simulation of $\gamma\gamma$ processes in JADE

Version 0

J. OLSSON

Introduction

Monte Carlo simulations of $\gamma\gamma$ physics processes, i.e. of the reactions

$$e^+e^- \to e^+e^-X \tag{1}$$

and the reactions contained in (1)

$$\gamma\gamma \to X \tag{2}$$

where X is a hadronic system (leptonic systems, i.e. pure QED processes, are not considered in the following), are done for several reasons:

a) Artificial events of the particular reaction under study can be used to optimize analysis cuts.

b) The artificial events can be used to determine the trigger efficiency for the particular reaction. The trigger efficiency is an essential part of the overall detection efficiency. If the latter is known, the observed number of events will provide a measurement of the cross section of (1).

c) The simulation automatically provides the $\gamma\gamma$ luminosity in some form. This number is needed to turn the cross section measurement of (1) into a corresponding measurement of the cross section for (2). The latter can then, in the case of X being a single resonance, be related to the radiative width of X, $\Gamma_{X\gamma\gamma}$.

In the JADE software for Monte Carlo simulation of reaction (1), two major steps can be distinguished, namely

1) Generation of 4-vector events of reaction (1), including 4-vectors of the decay products of X.

2) Simulation of the response of the JADE detector, by following each of the generated 4-vectors of the event through the JADE detector simulation program. The latter produces events in the same format as the real data events, i.e. in BOS format. These so called Monte Carlo events can be analysed in the same way as real data events.

In the following, each of these major steps will be shortly described. Several good works in the literature will be often referenced and the reader is recommended to get familiar with them, for the deeper understanding.

1. G. Bonneau, M. Gourdin and F. Martin, Nucl.Phys.**B54**(1973)573.

2. V.M. Budnev et al., Phys. Reports**15**(1975)181.

3. J. Field, Nucl. Phys**B168**(1980)477, Erratum Nucl.Phys.**B176**(1980)545.

4. M. Poppe, Intl.J.Mod.Phys.**1**(1986)545, DESY 86–014

5. J.D. Jackson, Nuovo Cim.**34**(1964)1644.

Generation of 4–vector events

The QED expression describing the density of virtual photons in reaction (1) is fairly complicated, although well understood (Budnev, Bonneau). In the early days of experimental $\gamma\gamma$ physics, various approximate (and simpler) formulas were used for its numerical calculation, based on the so called Equivalent Photon Approximation, EPA (Budnev, Field). Nowadays, the complexity of the full formula offers no problems, thanks to modern computer technology, and all large experiments doing $\gamma\gamma$ physics have developed sophisticated computer programs to perform the calculation of this density and to integrate it with the produced hadronic system. In the case of JADE, this work was done mostly by S. Kawabata, an earlier member of the group. His programs are based on a program developed by J. Vermaseren for the pure QED process

$$e^+e^- \rightarrow e^+e^- \; e^+e^- \tag{3}$$

or its analogues with μ pair or τ pair production. The method used is *Importance Sampling*; for a description of this method see e.g. the original write–up by S. Kawabata (also S. Kawabata, Comp.Phys.Comm.**41**(1986)127), or Poppe. The heart piece of these programs is the routine VEGAS, originally developed by P. Lepage. It was modified by Kawabata and renamed to BASIS (GRUND in some versions) or SPRING (QUELLE in some versions).

The numerical integration of (1) proceeds in at least 5 dimensions: the energies and the θ angles of the scattered electrons and also the relative ϕ angle of the electrons. If the mass of the produced system (usually called $W_{\gamma\gamma}$) is not fixed, then also this will be integrated over and the process is 6–dimensional. Even for modern computers this is CPU–time demanding. Kawabata therefore arranged the integration in such a way that the time consuming part i.e. the integration itself, is done only once. The tossing of 4-vector events, which is the final goal of the work, can then be done with great speed and repeated many times with new sets of random numbers. The following program descriptions therefore always come in pairs: one part is doing the integration with the integration results being written on a small file; the second part is a program which reads this integration file and uses its results for the 4–vector event generation.

Below several program examples are described in detail (they can be found on the source library F11OLS.JADEMC, which contains a collection of JCL– members for various $\gamma\gamma$ Monte Carlo jobs):

#SBRW70,#SBRW71: The first program performs the integration of reaction (1) with X being a broad resonance, e.g. the $f_2(1270)$. The second program performs the corresponding event generation.

#SBRW70 contains at the end the following data cards, for steering:

```
//*
//*  E          WS         WIDTH      W-MIN      W-MAX     ITT ICOND IFLAG ISPIN
//*(F10.4)    (F10.4)    (F10.4)    (F10.4)    (F10.4)  (I5) (I5) (I5)   (I5)
//*  FMAS1      FMAS2      FERMI
//*(F10.4)    (F10.4)    (F10.4)
//*
//GO.SYSIN    DD *
COMPUTE THE SINGLE BREIT-WIGNER SYS PROD. IN GG PROCESS
    17.3000     1.2740     0.1780     0.2700     3.0000   50    3    0    2
     0.1350     0.1350     1.0000
```

The input variables are now explained in detail:

E: The Beam Energy, in GeV.

WS: The Mass of the produced resonance X, in GeV. For a broad resonance, it should be the nominal mass.

WIDTH: The Width of the produced resonance X, in GeV. For a broad resonance, it should be the nominal width.

W-MIN: Lower end of the mass range of X. WMAX is the upper end of the range. In principal, the mass range is 0 – $2{\cdot}E_{\mathrm{beam}}$, neglecting electron masses. However, in most cases only a small part of this range is really needed, and it is very ineffective to integrate over the complete range. Besides, this may lead to numerical problems. In the example, the $f_2(1270)$ is integrated between the lower limit of two π° masses (its decay products) and the upper limit of 3 GeV, where the resonance is effectively zero again.

ITT: Number of iterations. The precision of the integration, normally 0.1%, is given by the internal variable BCC. If this precision is not reached, the program will end the integration after ITT iterations.

ICOND: This flag is used for steering cuts on the kinematic variables in the integration. This is sensible in some cases, e.g. if only tagged events are wanted (or only untagged). The actual cuts have to be coded for. In the given example, ICOND is a dummy. Note that cuts affecting the integration can only be done on the primary produced particles, i.e. the outgoing scattered electrons and the produced system X. Decay products of X appear only in the second step, when the 4-vector events are produced (see below).

IFLAG: This flag is used for telling the program to start from scratch or to continue with an already started integration. The integration program is clever enough to recognize if too little time is left to perform one more iteration and, in case of such impending time limit, writes the intermediate results onto the output result file. The user is informed by a print out, saying e.g. that integration ended with flag value 2. It is then possible to

resubmit the job, with IFLAG set to 2, and continue the integration. In this case, care has to be taken to define the output file correctly, since it is now assumed already existing and used first as input. When the integration has reached the level of IFLAG=3, the output file will be organized such that it can be used as input for the following 4-vector generation program. Thus it is not necessary to continue the integration to the ITT limit or to the specified precision, since it may happen that the program actually is unable, for numerical reasons, to obtain the wanted precision.

ISPIN: The spin of the produced resonance, in the case of $f_2(1270)$ it is 2.

FMAS1: Together with FMAS2 the masses of the decay products of the resonance, in this case both are π°. 2–body decay is assumed, for the parametrization of the resonance as a Breit–Wigner curve.

FERMI: A parameter used in the parametrization of the mass–dependent width of the resonance. More about this below.

The integrated function is given in the program by the FUNCTION F(X), where X is an array of random numbers supplied by the calling routine BASIS. F(X) calls LMFUNC, which provides the $\gamma\gamma$ luminosity, essentially formula (28) and (29d) in Bonneau. The subroutine BREIT gives the $\sigma_{\gamma\gamma}$, which is just a Breit–Wigner formula, (3.24) in Budnev. Later in F(X) some additional Q^2 dependence is added by ρ–poles. This can be replaced by the Q^2 dependence of GVDM, by use of the FUNCTION FGVDM; presently this is commented out.

Before proceeding with the description of the actual 4-vector event generation, some comments on the code of the subroutine BREIT should be given. The formula (3.24) in Budnev is

$$\sigma_{\gamma\gamma} = 8\pi(2J+1)\cdot\Gamma_{X\gamma\gamma}\cdot\frac{\Gamma}{(W^2-m_X^2)^2+\Gamma^2 m_X^2} \tag{4}$$

Here J is the spin and Γ the total width of the resonance. The total width is normally parametrized as a mass-dependent function for broad resonances (see Jackson):

$$\Gamma = \Gamma_0\cdot\left(\frac{p}{p_0}\right)^{(2J+1)} \tag{5}$$

Γ_0 is the nominal width of the resonance, p_0 is the momentum of the decay products (2–body decay!) at the nominal mass, m_X, and p is the corresponding momentum at the actual mass W. The parametrization (5) leads to an asymmetric shape, the more so with higher spin J, and normally an additional damping factor is needed to supress the Breit–Wigner curve at higher masses. This extra damping is usually parametrized as an Angular Momentum Barrier Penetration Factor (see Blatt&Weisskopf, Theoretical Nuclear Physics), sometimes also called Decay Form–factor. It has the form

$$\begin{aligned} \text{Spin 0:}\quad & D(x) = 1 \\ \text{Spin 1:}\quad & D(x) = 1 + x^2 \\ \text{Spin 2:}\quad & D(x) = 9 + 3\cdot x^2 + x^4 \end{aligned} \tag{6}$$

with $x = p\cdot r$. p is the momentum and r is a length parameter, usually taken as 1 Fermi. x_0 is the corresponding quantity at the nominal mass. The total width then takes the form

$$\Gamma = \Gamma_0\cdot\left(\frac{p}{p_0}\right)^{(2J+1)}\cdot\frac{D(x_0)}{D(x)} \tag{7}$$

The above parametrizations are empirical and the theoretical foundations can be discussed. For very broad resonances, like the ρ, still more empirical shaping functions can be added (see Jackson). Moreover, the whole formula (4) is approximative, since it is written for the limit of real photons (Q^2=0); More accurate descriptions can be found in e.g. Poppe. See also below the discussion on very narrow resonances.

The second JCL–member, #SBRW71, is very similar to #SBRW70. The steering data cards at the end are the following:

```
//*
//*  E          WS        WIDTH      W-MIN      W-MAX   ICOND IXTYP ISPIN
//*(F10.4)   (F10.4)   (F10.4)   (F10.4)   (F10.4)  (I5) (I5)   (I5)
//*  FMAS1      FMAS2     FERMI     NR.EV
//*(F10.4)   (F10.4)   (F10.4)   (I10)
//*
//GO.SYSIN    DD *
COMPUTE THE SINGLE BREIT-WIGNER SYS PROD. IN GG PROCESS
   17.3000    1.2740    0.1780    0.2700    3.0000    0   20    2
    0.1350    0.1350    1.0000    999999
```

Most of these are the same as for #SBRW70 and it is the responsibility of the user to take care that they are really the same as used in the integration. In principle this could have been arranged in a safer way, but for historical reasons it was not. Two variables are new:

IXTYP: Every resonance that is generated has its type number, in the case of $f_2(1270)$ it is 20. The number is used to steer the selection of decay routines, to be described below.

NR.EV: The number of 4-vector events one wants to generate. Normally set to a large number, since the generation of events will be stopped by TIME LIMIT or disc overflow.

There are two or more steering variables to be set, but for historical reasons they are set directly in the FORTRAN code of the MAIN program, which is the first in the JCL member:

LIMRND: Random number seed. A new seed will generate a new set of 4–vector events. *This number should be an odd number.*

KTYPE: This is an array, sitting in COMMON /CDKTYP/. Each element corresponds to a particle type, IXTYP. The value of a given element decides the decay mode of the corresponding resonance. In the example, $f_2(1270)$ decays into $\pi^\circ\pi^\circ$, and each π° decays into 2γ. More about this below in the description of SAGE.

The function F(X) and its called subroutines are identical to those in #SBRW70. BASIS is however now replaced by the slightly different version SPRING, which produces 4-vector events of weight 1. An event is first only produced as the final state e^+e^- X, and in a second stage X is decayed into stable particles by a call to the subroutine DECAYS. The latter routine, which calls a number of other decay routines, is initialized by the previous call to subroutine MTSTRT. The subroutine ROTAT3 provides a random rotation in ϕ, since only the relative ϕ between the electrons is considered in the integration and generation, and the extra overall rotation in the LAB–system has to be added. WRIT4V finally writes the final 4–vector event out in the special format, the so called CPROD–format (named after the COMMON/CPROD/) adopted in the JADE set–up.

The decay routines use the program package SAGE, which produces decay distributions in LIPS (Lorentz Invariant Phase Space). SAGE was written by Jerry Friedman and a long

write–up is available. All the SAGE routines, as well as the special JADE routines for decays, which call SAGE routines, are located on the source library F11OLS.JADEMC2 (member MTSTRT); the load versions are found on F11OLS.JADELD. The original SAGE routines have been copied from the original library F11BAR.EVENTGEN.S. It is worth remarking that this set of SAGE routines is very old, it was used already in 1974-75 and the import route from SLAC to DESY is not quite clear. In the winter 1986-87 the most actual version used presently at SLAC (where the author of SAGE also resides) was imported and checked very carefully against the old version used in JADE, and no disagreement was found, apart from esthetic changes in the code.

No detailed description of SAGE will be given here, the interested reader is referred to the write–up and program examples. It is however probably useful to give here a summary of the JADE specific details:

COMMON/CPROD/

The 4-vector events are kept in a special format, for historical reasons not a BOS–bank. It is given by the following MACRO, sitting on the source library F22ELS.JMC.S, which is the standard JADE Monte Carlo library:

```
C ------------------ MACRO CPROD -----------------
C                                        4-VECTOR FORMAT FOR JADE
C
       COMMON/CPROD/ NEV,BEAM,PT,PHI,THETA,IFLAVR,
     *         NP,NC,NN,PP(4,500),XM(500),JCH(500),JTP(500),JP(500,2),
     *         NF,NCF,NNF,PF(4,300),XMF(300),ICF(300),ITF(300),
     *         PSTRT(3,300)
C
C ---------------END MACRO CPROD -----------------
```

The variables are as follows:

NEV: Generated event nr, starting with 0. After detector simulation, this is the number in HEAD bank word 11.

BEAM: The beam energy.

PT: This and the following PHI,THETA,IFLAVR have no meaning in $\gamma\gamma$ physics and are not set.

NP: Nr of produced particles, before subroutine DECAYS has been called. In our case this is 3, i.e. e^+e^- X. Of course, NP = NC + NN.

NC: Nr of charged produced particles. In our case 2, i.e. e^+e^-.

NN: Nr of neutral produced particles. In our case 1, i.e. X.

PP: 4–vector array of the produced particles 1–NP.

XM: Masses of the produced particles 1–NP.

JCH: Masses of the produced particles 1–NP.

JTP: Types of the produced particles 1–NP.

JP: Pointers of the produced particles 1–NP, referring to decay daughters in the final particles.

NF: Nr of final particles, after subroutine DECAYS has been called. In our case this is 6, i.e. $e^+e^-$and 4 γ. Note that stable produced particles (e^+e^-) are just transferred to the final particles. Again, NF = NCF + NNF.

NCF: Nr of final charged particles, in our case just the 2 electrons.

NNF: Nr of final neutral particles, in our case 4 photons.

PF: Analogue of PP for the final particles.

XMF: Analogue of XM for the final particles.

ICF: Analogue of JCH for the final particles.

ITF: Analogue of JTP for the final particles.

PSTRT: x-y-z of the start point for a final particle, e.g. this could be the decay point in space of a K^0.

PARTICLE TYPES

These are partly (types 1–12) given already in Jade Computer Note 10, but have been further extended for $\gamma\gamma$ physics use. The original set have been used for general Monte Carlo work in JADE and other special areas may have obtained their own extensions of this numbering scheme; there is no overall standardization of this in JADE. Moreover, other numbering schemes are also used, e.g. the one in the LUND Monte Carlo programs. The numbering used in $\gamma\gamma$ physics can be found in the Commented Header of the main routine DECAYS, which at present is the following:

```
C      LAST MODIFICATION:  10.11.1985     J.OLSSON
C
C    LIST OF PARTICLE TYPES:
C      0   NEUTRINO                 20    F0(1270)
C      1   PHOTON                   21    A2(1310)
C      2   ELECTRON                 22    E(1420)
C      3   MUON                     23    U(2981)     (ETAC)
C      4   PION                     24    F1(1515)
C      5   K                        25    S*(980)
C      6   NUKLEON                  26    JOTA(1440)
C      7   PHI                      27    DELTA(981)
C      8   ETA                      28    X(2100)
C      9   ETAPRIME                 29    TAULEPTON (1784)
C     10   OMEGA                    30    PI+PI- SYSTEM (BORN)
C     11   KSTAR(890)               31    ZERVAS HEAVY PSEUDOSCALAR
C     12   RHO                      32    BARYON-ANTIBARYON SYSTEM
C     13   CHI(3.55)                33    BARYON-ANTIBARYON-PIO SYSTEM
C     14   J/PSI
C     15   A3(1680)
C     16..19   NOT USED             50    KO  (USES KTYPE(5) FOR DECAY!)
```

Each of the particles (i.e. the unstable ones) has its own decay routine, with various options for decay modes. The most important are listed below:

pion: Subroutine PI0DK. Only π° decays in the 4-vector event generation, the charged π's are considered stable and may decay in the detector simulation. For the π°, 2 decay modes are simulated:

```
MODE 0:  GAMMA GAMMA              MODE 1:    GAMMA E+E-
```

ϕ: Subroutine PHIDK. The following decay modes are simulated:

```
MODE 0:  KL, KS                   MODE 1:    ETA GAMMA
MODE 2:  K+, K-                   MODE 3:    PI+ PI- PIO
```

$\eta(549)$: Subroutine etaDK. The following decay modes are available:

```
MODE 0:  GAMMA GAMMA              MODE 1:    PIO PIO PIO
MODE 2:  PI+ PI- PIO              MODE 3:    PI+ PI- GAMMA
MODE 4:  PIO GAMMA GAMMA
```

$\eta'(958)$: Subroutine ETPRDK. The following decay modes are available:

```
MODE 0:  GAMMA GAMMA              MODE 1:    RHO GAMMA
MODE 2:  PI+ PI- ETA              MODE 3:    PIO PIO ETA
MODE 4:  OMEGA GAMMA              MODE 5:    PI+ PI- GAMMA
```

ω: Subroutine OMEGDK. The following decay modes are available:

```
MODE 0:  PIO GAMMA                MODE 1:    PI+ PI- PIO
MODE 2:  PI+ PI-
```

$K^*(892)$: Subroutine KSTDK decays the neutral K*(892) with decay modes:

```
MODE 0:  PIO K(SHORT)             MODE 1:    PIO K(LONG)
MODE 2:  K+- PI-+
```

The charged K*(892) is decayed with subroutine CKSTDK and modes:

```
MODE 0:  PIO K+-                  MODE 1:    PI+- K(LONG)
MODE 2:  PI+- K(SHORT)
```

ρ: Subroutine RHODK with the single mode $\pi^+\pi^-$, or $\pi^\pm\pi^0$, depending on charge.

$f_2(1270)$: Subroutine F0DK with the following decay modes:

```
MODE 0:  PIO PIO                  MODE 1:    PI+ PI-
MODE 2:  PI+ PI- PI+ PI-          MODE 3:    K+ K-
MODE 4:  PI+ PI- ETA              MODE 5:    PI+ PI- GAMMA
MODE 6:  PIO PIO GAMMA            MODE 5:    KO(SHORT) KO(SHORT)
MODE 8:  PIO PIO PI+ PI-
```

$a_2(1320)$: Subroutine A2DK with the following decay modes:

```
MODE 0:  RHO+- PI-+               MODE 1:    ETA PIO
MODE 2:  OMEGA PIO                MODE 3:    K+ K- (OR KOS KOS)
MODE 4:  PI+ PI- PIO
```

E(1420): Subroutine EDECAY with the following decay modes:

```
MODE 0:  KO(SHORT) K+- PI-+       MODE 1:    K+ K- PIO
MODE 2:  2/3 ETA PI+ PI-, 1/3 ETA PIO PIO
```

$\eta_c(2981)$: Subroutine UDECAY with the following decay modes:

```
MODE 0:  KO(SHORT) K+- PI-+       MODE 1:    K+ K- PIO
MODE 2:  ETA PI+ PI-              MODE 3:    ETA PIO PIO
MODE 4:  PI+ PI- PI+ PI-          MODE 5:    PI+ PIO PI- PIO
```

```
          MODE 6:   PI+ PI- GAMMA          MODE 7:    ETAPRIME PI+ PI-
          MODE 8:   K+ K- K+ K-            MODE 9:    K+ K- KOS KOS
```

$f_2'(1525)$: Subroutine F1DK with the following decay modes:

```
          MODE 0:   PIO PIO                MODE 1:    PI+ PI-
          MODE 2:   K+ K-                  MODE 3:    KOS KOS
          MODE 4:   PI+ PI- ETA
```

$S^*(975)$: Subroutine STARDK with the following decay modes:

```
          MODE 0:   PIO PIO                MODE 1:    PI+ PI-
```

$\eta(1440)$: Subroutine JOTADK with the following decay modes:

```
          MODE 0:   DELTA PIO
```

$\delta(981)$: Subroutine DELTDK with the following decay modes:

```
          MODE 0:   K+ K-                  MODE 1:    KOS KOS
          MODE 2:   ETA PIO
```

K_S^0: Subroutine K0DK with the following decay modes:

```
          MODE 0:   PIO PIO                MODE 1:    PI+ PI-
```

Simulation of the JADE Detector

The 4–vector events written on the output file by #SBRW71in the program example described above, serve as input to the next major step, namely the detector simulation, or in jargon "tracking". Contrary to the above, this program package is well standardised in JADE and also well described. The following Jade Computer Notes deal with tracking: 54,67,69,70,72,86,87 and 87 addendum.
Unfortunately, some are a bit outdated and none is really written for the newcomer. A simple outline of the tracking program is therefore given below. The example is taken from the authors specially adapted program, but the main flow is the same as in all Monte Carlo simulations in JADE:

```
      COMMON / BCS / IW(40000)
C
      COMMON / CIEVS  / KIEV,IEVMIN,IEVMAX
C
C           IEVMIN : FIRST EVENT TO BE READ FROM INPUT FILE
C           IEVMAX : LAST  EVENT TO BE READ FROM INPUT FILE
C
C                   ==== > DEFAULT IEVMIN = 1
C                                  IEVMAX = 9999999
C
      INTEGER*2 HDATE
      COMMON / TODAY / HDATE(6)
C
      COMMON/CFLAG/LFLAG(10)
      LOGICAL * 1 LFLAG
C
C           LFLAG(1) = SMEAR GAMMA AND ELECTRON ENERGIES
C           LFLAG(2) = GAMMA CONVERSION IN OUTER TANK AND COIL (TRKGAM)
C           LFLAG(3) = ABSORPTION LOSSES
```

```
C           LFLAG(4) = 3 DIM SHOWER PROFILE FIT TO EGS CODE
C           LFLAG(5) = .TRUE.   -->  WITH VERTEX CHAMBER TRACKING
C                    = .FALSE.  -->  WITHOUT VERTEX CHAMBER TRACKING
C                                    BUT OLD BEAM PIPE GEOMETRY AND
C                                  BEAM PIPE COUNTERS (BEFORE MAI 84)
C           LFLAG(6) = 3 DIM TOKYO SHOWER PROGRAM, JADE NOTE 20A
C
C          ===> DEFAULT .TRUE.,.FALSE.,.TRUE.,.FALSE.,.FALSE.,.FALSE.
C
C    NOTE: WITH THE USE OF TOKYO SHOWER PROGRAM, LFLAG(1) AND LFLAG(3)
C          ARE NOT USED. HOWEVER, THEY SHOULD BE TRUE TO ENSURE THAT THE
C          ALGN BANK IS PROPERLY MARKED FOR LATER ENERGY CORRECTION..
C          LFLAG(2) IS TRULY REDUNDANT
C          LFLAG(4) SHOULD BE FALSE, ALTHOUGH TRLGL IS ONLY CALLED FOR
C                   HADRONS AND MUONS
      LOGICAL*1 TEXT
      DIMENSION TEXT(72)
C
      LFLAG(1) = .TRUE.
      LFLAG(2) = .FALSE.
      LFLAG(3) = .TRUE.
      LFLAG(4) = .FALSE.
      LFLAG(5) = .TRUE.
      LFLAG(6) = .TRUE.
C
      WRITE(6,521) (LFLAG(I),I=1,6)
521   FORMAT(' LFLAG:  ',6I3)
C
C   INIT OF TODAY
C
      HDATE(1) = 1
      HDATE(2) = 1
      HDATE(3) = 1
      HDATE(4) = 15
      HDATE(5) = 6
      HDATE(6) = 1986
C
      WRITE(6,2201) HDATE
2201  FORMAT('  DATE: ',6I6)
C                                Initialize BOS
      CALL BINT( 40000,15000,500, 0 )
C
C    READ IN:    EVENT READING LIMITS    ievmin ievmax
C
7608  FORMAT(72A1)
7610  FORMAT(7I10)
7508  FORMAT(' ',72A1)
7510  FORMAT(' ',7I10)
```

```
      READ 7608, TEXT
      WRITE(6,7508) TEXT
      READ 7610, IEVMIN,IEVMAX
      WRITE(6,7510) IEVMIN,IEVMAX
C                              main tracking routine
      CALL MCJADE( 0,  5 )
C                              BOS statistics
      CALL BSTA
      CALL PALL
      STOP
      END
```

For the beginner, nothing in this MAIN program really needs explanation. HDATE is an array, which setting determines the "date" of the simulated detector; this is important, when one goes into details of triggering efficiencies or thresholds of SF5/SF6 Lead glass, etc.. Changing of the logical flags LFLAG needs expertise and the default is recommended as above. An important feature is the ability to stear the start and end event. Since parts of the program are rather slow, and the input file with the 4-vector events contains many thousand events, one may need many shorter jobs, or several longer ones, to work through a larger sample. It is convenient to be able to submit many jobs at the same time, each being set up for a particular part of the input file. If the output file is a disc file, which is the normal case, one has to remember that the ready tracked events are full BOS-format events which take much bigger space on disc than the input 4–vector events.

The main difference between the above MAIN program and the standard one (which is situated on F22ELS.JMC.S/L) is the use of LFLAG(6), to invoke the use of the full lead glass shower program from TOKYO. The corresponding version of the main routine, MCJADE, is situated on F11OLS.JADE56.S/L, where also all other routines involved in this shower program package are situated. They will be fully described in a forthcoming Jade Computer Note.

The library F11OLS.ETAP.S contains 2 program examples, with which tracking can be started: #LINKMC and JBTJ20D. The former creates a linked module of the tracking program, which is then used by JBTJ20D. The many entries in the INCLUDE statements are historical and will be made standard as soon as the mentioned Jade Computer Note is made available. Note that in the tracking job, JBTJ20D, the input file is by default on unit 3, the output on unit 2; this is just the opposite of the normal SUPERV standard. It is a nice example of the rich individualism in HEP collaborations.

The output events are in BOS format, they are by all means real JADE events, although they are in fact raw data events, and need all the standard analysis treatments of pattern recognition and cluster analysis. This can be performed by any standard job, e.g. #TG04. They can also be directly looked at in the JADEZ graphics program. In the latter case, note that the command TRUE will display the original 4-vectors, which are kept in a special BOS bank with name VECT.