Computer Physics Communications 41 (1986) 127–153
North-Holland, Amsterdam

A NEW MONTE CARLO EVENT GENERATOR FOR HIGH ENERGY PHYSICS

S. KAWABATA

National Laboratory for High Energy Physics, Tsukuba, Ibaraki, Japan

Received 3 August 1985; in revised form 7 December 1985

PROGRAM SUMMARY

Title of program: BASES/SPRING

Catalogue number: AAFW

Program obtainable from: CPC Program Library, Queen's University of Belfast, N. Ireland (see application form in this issue)

Computer for which the program is designed and others on which it is operable: FACOM, HITAC, IBM and others with a FORTRAN77 compiler

Computer: FACOM-M382, HITAC-M280; *Installation*: National Laboratory for High Energy Physics (KEK), Tsukuba, Ibaraki, Japan

Operating system: OSIV/F4 MSP (FACOM), VOS3 (HITAC)

Programming language used: FORTRAN77

High speed storage required: 145 Kwords each for BASES and SPRING

No. of bits in a word: 32

Peripherals used: terminal or card read for input, disc and line printer for output

No. of lines in combined program and test deck: 2750

Keywords: Multidimensional numerical integration, Monte Carlo simulation, event generation and four momentum vector generation

Nature of physical problem
In high energy physics experiments, it is necessary to generate events by the Monte Carlo method for the processes of interest in order to know the detection efficiencies. It is desirable to have such a program that can generate events efficiently once we have the differential cross section of any process. It will be a very powerful tool for the high energy physics experimentalist.

Method of solution
A pair of program packages BASES and SPRING is developed for solving such a problem. By BASES, the differential cross section is integrated and a probability distribution is produced. By SPRING, four momentum vectors of events are generated according to the probability distribution made by BASES.

Typical running time
The running time depends especially on the complexity of the function program which gives the differential cross section for the process. If we take the process $e^+e^- \to (Z^0) \to \nu\bar{\nu}\gamma$ as an example, the integration takes 11 s and the generation rate is 4500 events/s on the FACOM M382 computer.

LONG WRITE-UP

A Monte Carlo event generator designed to be suitable for high energy particle physics is developed, by which events of any processes with 10 variables in maximum are generated efficiently. The program consists of two sets of packages, one is a numerical integration package BASES preparing a probability distribution and the other is an event generation package SPRING. The central part of the paper describes how to use the program packages in details.

1. Introduction

In high energy physics, when we get physical results from experimental raw data, it is necessary to know detection efficiencies for the processes of interest. We generate many events, consisting of the four momentum vector of particles produced in these specific processes by the Monte Carlo method, pass them through a detector simulation program to produce artificial raw data and analyze the data by the same analysis program as used for the real data.

For generation of Monte Carlo events there is no general method suitable for any process. It is always necessary but not easy to find a method appropriate to the process being considered. For example, efficient generation of events for radiative Bhabha scattering or that for two photon processes is difficult, because these processes have many sharp peaks due to singularities in their differential cross sections. To generate events by the Monte Carlo method, we normally look for a function, whose value is always greater than that of the differential cross section and whose generating function has an inverse function. When we find fortunately such a function, we can generate the events efficiently. The generation efficiency depends on the difference between values of the function and the differential cross section. When we fail to find any such a function, we are forced to be satisfied with lower efficiency and waste much CPU time in some cases.

A pair of program packages BASES and SPRING has been developed to solve this problem. Roughly speaking, the method is to construct the generating function by a histogram with *variable bin size*, which is determined by the height of the differential cross section. Generation proceeds with the following two steps:

(a) By BASES, a probability distribution (generating function) is calculated by integrating the differential cross section over the phase space and saved in a file with parameters needed in the next step.

(b) By SPRING, events are generated by the Monte Carlo method according to the above distribution.

The numerical integration program package BASES contains also several facilities, as described in section 4.3, which are helpful for the user who wants to integrate singular functions. The integration method we adopt, is the algorithm of VEGAS [1], devised by Lepage, which has been proved to be very useful for integration of singular functions appearing in the calculation of the radiative QED processes.

The next section contains a brief description of the integration method. In section 3 the generation method is presented. In sections 4 and 5, how to use these program packages is explained with an example.

2. Integration method of BASES

We adopt the integration method of VEGAS. Since a detailed description of the algorithm is given in ref. [1], we review it here briefly.

For simplicity, we consider the following one-dimensional integral as an example:

$$I = \int_0^1 f(x)\,\mathrm{d}x, \tag{1}$$

where the integrand $f(x)$ must be *non-negative* for the sake of event generation.

We divide the integration volume (0, 1) into N_g regions and each of them into m subregions. Namely, the whole integration volume consists of N_d subregions, i.e.

$$N_d = N_g m \quad \text{and} \quad \sum_{j=1}^{N_d} \Delta x_j = 1, \tag{2}$$

where Δx_j represents the size of the subregion j. We define the probability density p_j for the subregion j as

$$p_j = \frac{1}{N_d\, \Delta x_j} \quad \text{and} \quad \sum_{j=1}^{N_d} p_j\, \Delta x_j = 1. \tag{3}$$

We sample N_{pg} points in each region to estimate the integral, such that the total number of sampling is

$$N_{call} = N_{pg} N_g. \tag{4}$$

The estimated value of the integral I is given by

$$S = \frac{1}{N_{call}} \sum_{i=1}^{N_{call}} \frac{f(x_i)}{p_j} = \frac{1}{N_{call}} \sum_{i=1}^{N_{call}} f(x_i) N_d\, \Delta x_j, \tag{5}$$

where the subregion j is that region which contains x_i. The variance σ^2 of the estimate S is given by

$$\sigma^2 = \frac{1}{N_g N_{pg}^2 (N_{pg} - 1)} \left\{ \sum_{l=1}^{N_g} \left[N_{pg} \sum_{k=1}^{N_{pg}} (F_l(x_k))^2 - \left(\sum_{k=1}^{N_{pg}} F_l(x_k) \right)^2 \right] \right\}, \tag{6}$$

where

$$F_l(x_k) = f(x_k)\, \Delta x_j N_d \tag{7}$$

and $x_k \in$ subregion $j \subset$ region l.

To obtain the final result of the integration, BASES makes n estimastes of the integral $\{S_\alpha\}_{\alpha=1}^{n}$, each of which is calculated by the above method. These n estimates are combined to give a cumulative estimate $\bar{S}$:

$$I \approx \bar{S} = \sigma^2 \sum_{\alpha=1}^{n} \frac{S_\alpha}{\sigma_\alpha^2} \tag{8}$$

and

$$\frac{1}{\sigma^2} = \sum_{\alpha=1}^{n} \frac{1}{\sigma_\alpha^2}, \tag{9}$$

where σ_α^2 is a variance of the αth estimate S_α.

In BASES, the density p_j, namely Δx_j, is modified so as to minimize σ_α^2 for each iteration of the estimation. In the first estimation, uniformly distributed random points are chosen (i.e., in determining S_1). The information about the behavior of $f(x)$ obtained in this first sampling is used to define a new density p_j which reduces σ_α^2 in the next iteration of the estimation for S_2. After each subsequent iteration, the density p_j is again refined for the next one. Namely, sizes of subregions Δx_j are adjusted after each iteraction.

In order to prepare the probability distribution for the event generation, the following procedure is taken:

(1) Grid defining step
To determine the optimum density p_j (i.e., sizes of subregions Δx_j), several iterations of the estimastion are performed until the estimated error given by eq. (9) becomes less than the specified value, or the number of iterations reaches its upper limit or the fluctuation of the accuracy becomes less than 1%.

(2) Accumulation step
By using the grids optimized by step (1), iterations of the estimation are carried out until the estimated error becomes less than the required value or the number of iterations reaches its given upper limit. In this step the probability distribution is calculated.

(3) Resulting probability distribution is saved in a disc file.

3. Method of event generation

In the N dimensional integration, we have N independent variables. We divide a region spanned by each variable into N_g regions and each of those N_g regions into m subregions. Therefore, the whole integration volume is divided into $(N_g)^N$ hypercubes by a rectangular grid of N_g subdivisions along each axis. A further subdivision defines a new grid m times finer on each axis and results in $(N_g m)^N$ sub-hypercubes. Each hypercube has m^N sub-hypercubes.

The probability distribution is prepared by BASES. By this we mean that both average and maximum values of probabilities

$$F(x_1, x_2, \ldots, x_N) = f(x_1, x_2, \ldots, x_N) N_d^N \prod_{i=1}^{N} \Delta x_i, \tag{10}$$

for each hypercube, as well as sampling intervals both for the hypercube and the sub-hypercube, are saved in a disc file.

For generation, a hypercube is chosen according to the average probability and a point in a sub-hypercube in the hypercube is sampled with the density p_j. $F(x_1,\ldots,x_N)$ is calculated at the point and its value is compared with the maximum value of probabilities $F_{\max} = \max [F(x_1,\ldots,x_N)]$ of the hypercube. If $F/F_{\max}$ is greater than a uniform random number between 0 and 1, then this trial is accepted as a generated event. If not, a new sub-hypercube is sampled and tried until it succeeds.

If the probability $F(x_1,\ldots,x_N)$ is almost uniform in each hypercube, the generation efficiency is close to 100%. This is the reason why we took VEGAS's algorithm to optimize the size of subregion so as to make the probability uniform. However, the actual probability is not always uniform resulting in a long loop of sampling a sub-hypercube in some case. To avoid this loop, SPRING has a user controllable parameter defining the maximum number of trials for one event generation. If the number of trials reaches the specified maximum, this loop is forced to terminate and a new hypercube is sampled for the next generation. If this happens so many times in a job, however, generated events cannot reproduce the behavior of the differential cross section. This situation occurs when singularities of the differential cross section are not sensed appropriately in the grid defining step. It is necessary to submit BASES again with more sampling points.

4. Program package BASES

4.1. Flow of the integration Job

As mentioned in section 2, the integration consists of two steps, i.e., the grid defining step and the accumulation step. In the grid defining step, grids are optimized by several iterations of the estimate. After defining the grid, the accumulation step starts where iterations of the estimate are carried out with the fixed grids. For both steps, if the accuracy of the integration reaches a required value or the number of iterations reaches the specified maximum, the integration is terminated. In the grid defining step, if *fluctuation of the accuracy* becomes less than 1%, the integration is also terminated. The maximum iteration numbers, ITMX1 for the grid defining step and ITMX2 for the accumulation step, can be given by the user in the user initialization routine USERIN (default numbers of ITMX1 and ITMX2 are 15 and 100, respectively). After termination of the integration, the rsults (e.g. integrated value and required histograms) are printed out.

The integration step may take time, for instance, when the integrand needs a long computation. It might happen that a job is terminated before reaching the above normal termination due to lack of CPU time. To prevent this occurrence, the routine watches the remaining CPU time and when it is not enough for the next iteration, all temporary information is stored on disc before the job is ended. The integration can be continued by submitting next job. We define an input flag IFLAG as follows:

IFLAG = 0: First trial of the grid defining step,
= 1: First trial of the accumulation step,
= 2: Continuation of the grid defining step,
= 3: Continuation of the accumulation step.

At the beginning of an integration job, IFLAG = 0 should be set. If the job is terminated due to lack of CPU time, the input flag for the next job is printed out. When the next job is submitted with this flag, the integration is continued further. The flow of the integration job is shown in fig. 1.

4.2. Program structure and its components

Program structure of the integration package BASES is shown in fig. 2. The program components of BASES, which concern users, are briefly described in the following. An explanation of the other components is given in appendix F. The routines marked by ** should be prepared by user.

MAIN routine **
The main routine is used just for BASES to know

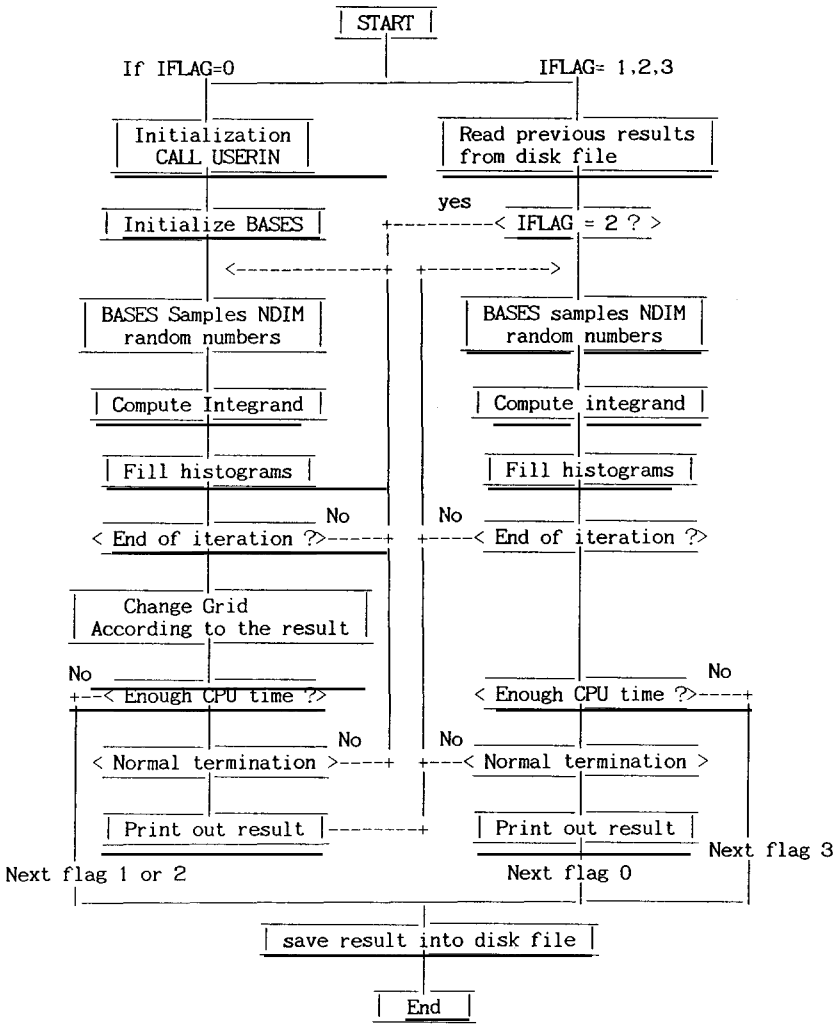


Fig. 1. Flow of the integration job.

the name of the integrand function program prepared by user. An example of the main routine is shown in appendix A, where the name of the function program is FUNC.

SUBROUTINE BSMAIN (FUNC)

Purpose: To control the flow of program package BASES according to the input flag IFLAG and to set default parameters. This is called by main program.

FUNC =: Name of integrand function.

SUBROUTINE USERIN **

Purpose: To initialize parameters in a labelled common /BASE1/ and user's function and to define histograms by XHINIT and DHINIT.

Comment: This routine is called by BSMAIN at the beginning of the integration job.

SUBROUTINE USROUT **

Purpose: To print the result in the format written by user.

Comment: This routine is called at the end of accumulation step, only when the print flag is set equal to zero.

FUNCTION DRN (IQ)

Purpose: To get double precision uniform random number.

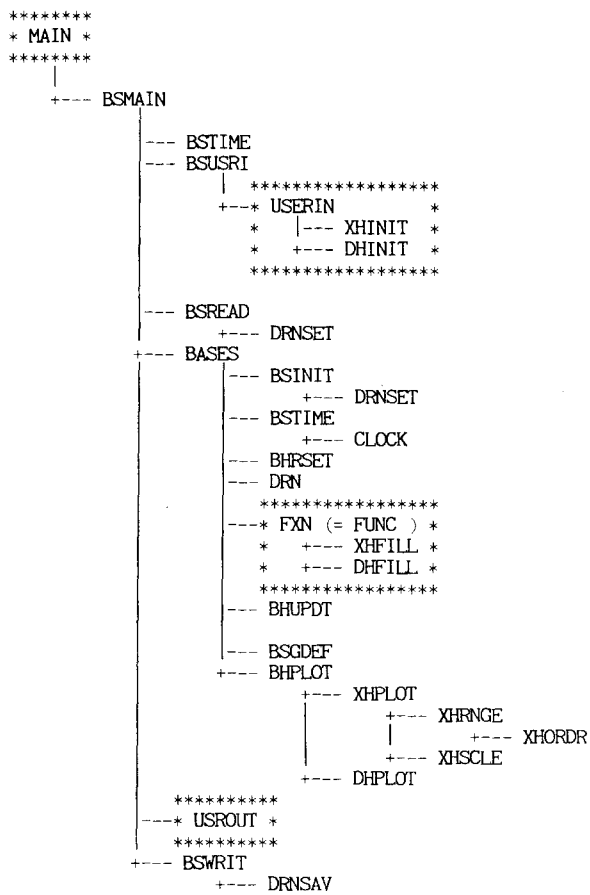


Fig. 2. Program structure of BASES. The routines marked by * are to be prepared by user.

IQ =: Dummy argument.

FUNCTION FUNC (X) **

Purpose: To calculate the integrand.

Comment: This function is prepared by user. Any name can be given to this function, but the name must be declared in the main routine by an external statement. If histograms are required, histogram-filling routines XHFILL and DHFILL should be called in this function program.

X =: Real * 8 dimensional array of the independent variables.

FUNC =: Resultant value of the integrand.

SUBROUTINE XHINIT (ID, XMIN, XMAX, NBIN, TITLE)

Purpose: To initialize a histogram.

Comment: This routine should be called in USERIN, if a historgram is required.

ID =: Histogram identification number.

XMIN =: Lower boundary of the variable.

XMAX =: Upper boundary of the variable.

NBIN =: Number of bins of the histogram (maximum = 50).

TITLE =: Title of the histogram, which is a character string up to 62 characters.

SUBROUTINE DHINIT (ID, XMIN, XMAX, NXBIN, YMIN, YMAX, NYBIN, TITLE)

Purpose: To initialize a scatter plot.

Comment: This routine should be called in USERIN, if a scatter plot is required.

ID =: Scatter plot identification number.

XMIN =: Lower boundary for the horizontal axis.

XMAX =: Upper boundary for the horizontal axis.

NXBIN =: Number of bins for the horizontal axis (maximum = 50).

YMIN =: Lower boundary for the vertical axis.

YMAX =: Upper boundary for the vertical axis.

NYBIN =: Number of bins for the vertical axis (maximum = 50).

TITLE =: Title of the plot, which is a character string of up to 62 characters.

SUBROUTINE XHFILL (ID, X, FX)

Purpose: To fill the IDth histogram.

Comment: This routine should be called in the user function program. Before calling this routine, the routine XHINIT should be called with the same ID number.

ID =: Histogram identifcation number.

X =: Variable.

FX =: Value of the integrand.

SUBROUTINE DHFILL (ID, X, Y, FX)

Purpose: To fill the IDth scatter plot.

Comment: This routine should be called in the user function program. Before calling this routine, the routine DHINIT should be called with the same ID number.

ID =: Plot identification number.

X =: Variable for the horizontal axis.

Y =: Variable for the vertical axis.

FX =: Value of the integrand.

In order to prepare the routine USERIN, it is necessary to know the contents of the labelled commons /BASE1/ and /BASE2/. Some of these parameters for BASES should be set by user and transferred to the routine BASES. The contents of /BASE1/ and /BASE2/ are as follows:

COMMON/BASE1/XL(10), XU(10), NDIM, NOCUB, NTRIAL

COMMON/BASE2/ACC1, ACC2, ITMX1, ITMX2

XL(i): Lower boundary of ith integration variable.

XU(i): Upper boundary of ith integration variable.

NDIM: Dimension of the integration (up to 10).

NOCUB: Maximum number of hypercubes (default and maximum numbers are 6000 and 17000, respectively).

NTRIAL: Number of sampling points per hypercube (default is 2).

ACC1: Required accuracy in the grid defining step (default value is 0.2%).

ACC2: Required accuracy in the accumulation step (default value is 0.01%).

ITMX1: Maximum number of iterations in the grid defining step (default number is 15).

ITMX2: Maximum number of iterations in the accumulation step (default number is 100).

At least XL(i), XU(i) and NDIM should be given in the routine USERIN.

There are relations among the dimension of the integration NDIM, number of hypercubes N_{cube}, number of grids per axis N_{g} and maximum number of hypercubes NOCUB. The relations are

Table 1
Relation among NDIM, N_{g}, N_{d} and N_{cube}

NDIM	N_{g}	N_{d}	N_{cube}
1	25	50	25
2	25	50	625
3	18	36	5832
4	8	48	4096
5	5	50	3125
6	4	48	4096
7	3	48	2187
8	2	50	256
9	2	50	512
10	2	50	1024

$$N_{\text{cube}} = (N_{\text{g}})^{\text{NDIM}} \leq \text{NOCUB}(=6000) < 17000, \tag{11}$$

$$N_{\text{d}} = N_{\text{g}} m \leq 50 \quad \text{and} \quad 2 \leq m \leq 25. \tag{12}$$

The number N_{g} is defined by the largest of those which satisfy the above relations. Then the actual number of total sampling points is

$$N_{\text{call}} = (N_{\text{g}})^{\text{NDIM}} \text{NTRIAL}. \tag{13}$$

The numbers N_{cube} and N_{g} are automatically determined according to NDIM and NOCUB. Table 1 shows these numbers with the default value 6000 of NOCUB varying NDIM from 1 to 10. The only way to change the number of sampling points N_{call} is to change NTRIAL. The description of the other labelled commons is given in appendix F.

4.3. Use of bases

We take the process $e^+e^- \to (Z^0) \to \nu\bar{\nu}\gamma$ as an example through this article. The physical conditions for the integration are taken to be the same as those of ref. [2] (the e^+e^- center of mass energy $W = 105$ GeV and the polar angle cut for the final photon $165° > \theta\gamma > 15°$), except for the energy threshold of the photon $E\gamma > 1/W$ GeV.

4.3.1. Preparation for BASES

Before using BASES user should prepare JCL, main program, user initialization routine USERIN and integrand function.

JCL and Main program for FACOM machine at KEK are shown in appendix A. In a file BASES.LOAD, the load modules of the BASES package are stored. Input parameters are the loop control parameters L1 and L2, print option flag NPRINT, input flag IFLAG described in section 4.1 and CPU time limit CTIME (real number in unit of minute) of the job. As described in section 4.3.3, the loop parameters and print flag for standard use are

$$\text{L1} = \text{L2} = 1 \quad \text{and} \quad \text{NPRINT} = 5. \tag{15}$$

If the integration is not finished in this CPU time, the job is terminated after all information is saved in a file @BASES.DATA of the unit number 23.

An example of USERIN is shown in appendix

B, where labelled commons /BASE1/, /BASE2/ and /KINEM/ are included. In the common /KINEM/, constants necessary to calculate the integrand function are stored. The upper half part of this USERIN is used for setting common parameters and the lower half for initializing BASES parameters and histograms. Those BASES parameters which are not defined in this routine are set equal to relevant default values. In this example two histograms and one scatter plot are defined.

Appendix C shows an example of integrand function FUNC. After calculating the function, histogram filling routines XHFILL and DHFILL are called. It is noted that ID numbers of histograms and scatter plots should coincide with those of initialization calls.

4.3.2. Output from BASES

Output from the BASES is printed out on papers and saved in a disc file. The former output consists of the following items:

(1) At the beginning of the job, job parameters are printed out.

```
* * * * * * * * * * * * * * * *
* Input Parameters for this JOB *
*                               *
* Current Loop Count   =     1 *
* Maximum Loop Count=     1 *
* Print Flag           =     5 *
* JOB Input Flag       =     0 *
* CPU Time Limit       =2.000 *
*                               *
* * * * * * * * * * * * * * * *
```

When user's own parameters are printed out in USERIN, they appear before this list.

(2) Main parameters for BASES are printed out.

```
* * * * * * * * * * * * * * * * * * * * * * * * *
*   Input Parameters for BASES                    *
*                                                 *
*   NDIM  =     2                                 *
*   ITMX1 =    15 ITMX2     =  50                 *
*   ACC1  =     0.200 ACC2   =  0.500E-01         *
*   NCALL = 3125 ND          =  50                *
*   NG    =    25                                 *
*   XL(1) =     0.250000E-01                      *
*   XU(1) =    20.0000                            *
*   XL(2) =    -0.965926                          *
*   XU(2) =     0.965926                          *
*                                                 *
* * * * * * * * * * * * * * * * * * * * * * * * *
```

(3) Histogram which shows convergence behavior of the integration. Its contents are as follows:

IT = Iteration number.
Time = Current CPU time in seconds.
Eff. = Efficiency of the sampling points.
R–Neg = Percentage of sampling points at which the function is negative.
Result = Estimate of the integration upto the current iteration.
St-Dev = Standard deviation upto the current iteration.
T-Res = Estimate of the current iteration.
T-Acc = Accuracy of the integration for the current iteration in percentage.
Acc = Accuracy of the integration upto the current iteration in percentage.

The last value Acc is drawn in the histogram.

(4) Histograms defined by user
In our example two histograms are defined in USERIN. One of them is shown in the test run output, where histogram with " * " mark is given in linear scale and that with "○" mark in logaritmic scale.

(5) Scatter plot defined by user
Our example requires one scatter plot, shown in the test run output. With this plot we can easily see how singular points are distributed in the two dimensional plane. To display the height of the distribution we divide relative height in each two dimensional bin into ten grades from 1 to 9 and * . If a bin has non-zero positive value but less than grade 1, we show this bin with ".". If a bin has zero value, this bin is blank. If a bin has a negative value, this bin is indicated by "–". It should be noted that *the negative value of the integrand is taken into account for estimate of the integration*. But for the event generation, the negative value is not allowed.

(6) Print message at the job termination
If the job is terminated by the CPU time limit, the following message is given:

```
* * * * * * * CPU Time out * * * * * *
          Next Loop Count =  1
          Next Input Flag =  3
```

When the next job is submitted with the loop

count 1 and the input flag 3, the integration is continued further.

If the job is terminated by convergence of integration or by reaching the maximum iteration number, the following message is printed out.

```
* * * * * End of BASES Loop * * * * * *
      Max. Loop Count =   1
```

In each of the grid defining step or the accumulation step, output of the above six items is obtained. If the job is terminated with flag 3, then its output consists of two sets, one is from the grid defining step and the other from the accumulation step. The next job, submitted with the input flag 3, gives only one set.

4.3.3. Remarks on using BASES

The numerical integration program package BASES has the following facilities which are useful to integrate a singular function.

(1) Watching remaining CPU time in a job
When we integrate a new function, we do not know how much CPU time it needs. If a job is terminated by the CPU limit, no information can be obtained from the job. This is just a waste of CPU resource. To avoid this occurrence, BASES watches the remaining CPU time in the job and if it is too short for the next iteration, all temporary information is saved on disc. By using this information on disc, the next job can continue further the integration.

(2) Histogramming package
BASES has histogramming facilities in itself, which allows the user to draw 10 histograms and 10 scatter plots in maximum. The integration variables should be carefully chosen so as to converge the integration faster. Normally, phyical variables can be transformed to relevant integration variables by an appropriate one-to-one mapping. It may happen that the mapping is not unique, i.e., an integration variable corresponds to two or more fold values of a physical variable. When we want to take a histogram with respect to such a physical variable, special care should be taken. For example, if a value x_1 of an integration variable x corresponds to two values y_1 and y_2 of a physical variable y, a histogram for the variable y should be filled by calling XHFILL two times in the function program. An example is the following: Let FNX(Y) be a function and FUNC, the sum of F1 and F2, give the integrand value. YJACB1 and YJACB2 are Jacobians.

```
F1 = FNX(Y1) * YJACB1
CALL XHFILL(ID, Y1, F1)
F2 = FNX(Y2) * YJACB2
CALL XHFILL(ID, Y2, F2)
FUNC = F1 + F2
```

(3) Loop option
As an example, we take the integration of the cross section for the process $e^+e^- \to (Z^0) \to \nu\bar{\nu}\gamma$. When we want to know the CM energy dependence of the cross section, we must integrate the cross section at several energy points. The loop option is designed for such a case.
To use the loop option,
(a) change USERIN: Insert the following statements into the list of appendix B:

```
COMMON/LOOP0/LOOP
REAL * 8 WCM(6)
DATA WCM/40.0, 60.0, 70.0, 105.0, 150.0,
                260.0/
IF(LOOP.LE.0.OR.LOOP.GT.6)STOP
W = WCM(LOOP)
```

(b) change the loop parameters and print flag of the input parameters in JCL as follows:

```
1, 6    Current loop count, Max. loop count
1       Print flag
0       Input Job Flag
10.0    CPU time in minutes
```

(c) submit job: If the job is terminated due to lack of CPU time, the next loop count and input flag are given at the end of the job (see section 4.3.2). When the next job is submitted with these flags, the integration can be continued. As the print flag is 1

Table 2
Definition of the print flag

Print flag	What is printed out?
0	nothing
1	only the final result of the integration
2	final result and histograms at the end of the accumulation step
3	convergence behavior and histograms only for the accumulation step
4	convergence behavior both for the grid def. and accum. steps and histograms for the accumulation step
$\geqslant 5$	all

here, the only final results are given as a table. The meaning of the print flag is given in table 2.

(4) Scatter plots
By taking scatter plots in various combinations of variables, we can easily see how singular points distribute in each two variable plane. It should be noted that *if singular points distribute along a diagonal line on this plane, the integration will converge very slowly or will not converge*. This is attributed to the integration algorithm. If this happens, the integration variables should be changed to avoid such situation.

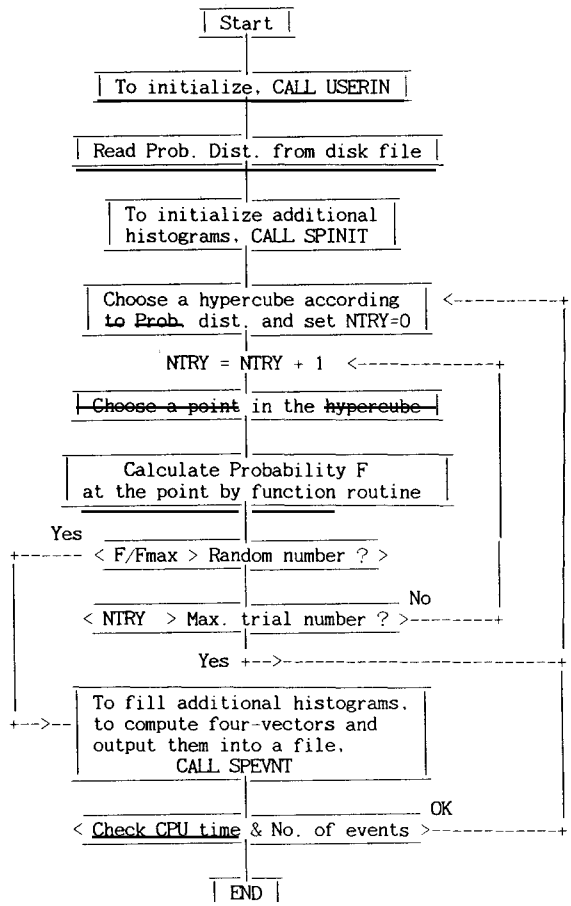


Fig. 3. Flow of the event generation job.

5. Program package SPRING

5.1. Flow of the generation job

Fig. 3 shows the flow of the generation job. To initialize parameters necessary for calculating the function, the routine USERIN is called at the beginning of the job, which should be exactly the same routine as used in BASES, and after that the routine BSREAD is called to read the results of BASES. The results of BASES include also histogram data. We call those histograms, which are created at BASES program, the "*original histogram*".

If the user wants to have additional histograms in this job, one should define them in the routine SPINIT. To make additional histograms, the user can use not only the internal histogram package XH * * * * and DH * * * * but also any external histogram packages like HANDYPACK and HBOOK.

The event generation loop proceeds in the following steps:

(1) A hypercube is chosen by uniform random number according to the probability distribution.

(2) A point in a sub-hypercube in the hypercube is sampled with the density p_j.

(3) Probability $F(x_1, \ldots, x_N)$ is calculated at the point and its value is compared with $F_{\max} = \max[F(x_1, \ldots, x_N)]$ of the hypercube.

(4) If $F/F_{\max}$ is greater than or equal to a uniform random number, this trial is accepted as a generating event. The routine SPEVNT is called

to handle the accepted event, which is prepared by the user. The SPEVNT calculates the four momentum vector (VECTOR(i)) of generated particles from kinematical variables and outputs them into a file. If additional histograms are required, they are filled in this routine. After call of SPEVNT, go to step (6).

(5) If the number of trials is less than the number defined by the user main program, go to step (2).

(6) If the residual CPU time is enough for next event generation and the number of generated events is less than the required one, then go to (1). Otherwise this job is terminated after calling SPTERM.

If additional histograms were defined by using an external histogram package, its printout routine should be called in the SPTERM.

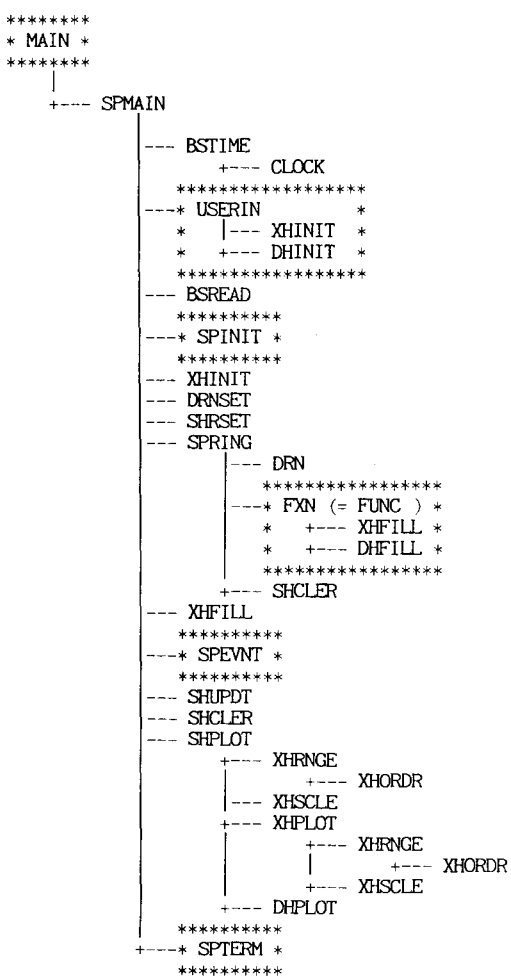


Fig. 4. Program structure of SPRING. The routines marked by * are to be prepared by user.

5.2. *Program structure and its components*

Program structure of the generation package SPRING is shown in fig. 4. The program components of SPRING, which concern users, are briefly described in the following. The other components are listed in appendix F. The routines marked by ** should be prepared by the user. Several routines listed below are identical to those in section 4.2.

MAIN ROUTINE **

The main routine is used for SPRING to know the name of the user function and the number of maximum trials for one event generation, which is prepared by user. An example is shown in appendix D.

SUBROUTINE SPMAIN (FUNC, MXTRY)

Purpose: To control the flow of program package SPRING.

FUNC =: Name of the user function.

MXTRY =: Maximum number of trials for one event generation.

SUBROUTINE USERIN **

Purpose: To initialize parameters for user function and to define the original histograms.

Comment: This routine must be identical with that used in BASES.

SUBROUTINE SPINIT **

Purpose: To initialize additional histograms.

Comment: If internal histogram package is used, ID number of these histograms should not be duplicated with the original histograms. An example is shown in appendix E. If an external histogram package is used, its library file should be included in JCL.

FUNCTION DRN (Q)

See section 4.2.

FUNCTION FUNC (X) **
See section 4.2.

SUBROUTINE XHINIT (ID, XMIN, XMAX, NBIN, TITLE)
Purpose: To initialize an additional histogram by using the internal histogram package.
Comment: For each additional histogram, this routine should be called in SPINIT. Description of the arguments is given in section 4.2.

SUBROUTINE DHINIT (ID, XMIN, XMAX, NXBINM, YMIN, YMAX, NYBIN, TITLE)
Purpose: To initialize an additional scatter plot by using the internal scatter plot package.
Comment: For each additional scatter plot, this routine should be called in SPINIT. Description of the arguments is given in section 4.2.

SUBROUTINE XHFILL (ID, X, FX)
Purpose: To fill the IDth histogram.
Comment: For each additional histogram, this routine should be called in SPEVNT. Description of the arguments is given in section 4.2.

SUBROUTINE DHFILL (ID, X, Y, FX)
Purpose: To fill the IDth scatter plot.
Comment: For each additional scatter plot, this routine should be called in SPEVNT. Description of the arguments is given in section 4.2.

SUBROUTINE SPEVNT **
Purpose: To calculate four momentum vectors of generated particles in the event and write them into a file. If additional histograms or scatter plots are required, the filling routines should be called here.

SUBROUTINE SPTERM **
Purpose: To terminate an external histogram package.

5.3. Use of SPRING

5.3.1. Preparation for SPRING

Before using SPRING the user should prepare JCL, main program, initialization routines USERIN and SPINIT, user function, event handling routine SPEVNT and termination routine SPTERM.

JCL on the FACOM machine and an example of the main routine are given in appendix D. In this main routine the maximum trials of one event generation is set to be 50. In our example, four momentum vectors of generated events are written into a file named @NUPAIR.EVENT.DATA.

The routine USERIN and function are already prepared for BASES and they must not be changed from those used in BASES. Since we required an additional histogram of the photon transverse energy, it is necessary to prepare SPINIT and SPEVNT. Examples of SPINIT and SPEVNT are shown in appendix E. In this SPEVNT four momentum vector of the generated photon is calculated, because the function is given by the formula which is already integrated over neutrino's phase space. Since we do not use any external histogram package, the routine SPTERM is just a dummy.

5.3.2. Output from SPRING

Output from the SPRING is as follows:

(1) Number of generated events

```
* * * * * * * * * * * * * * *
*                             *
*       END OF SPRING         *
* No. of Generated events     *
*           100 000           *
* Net CPU time for generation *
*           12.507 seconds    *
*                             *
* GO time                     *
*           22.077 seconds    *
*                             *
* * * * * * * * * * * * * * *
```

The net CPU time indicates the used CPU time only for the generation, which does not include, for instance, that for filling and printing histograms.

(2) Original histograms defined in BASES
Our example contains two original histograms. One of them is shown in the test run output. In these histograms the distribution of generated

events are indicated by "0" on the histogram of BASES result.

(3) Additional histograms from the internal histogram package
We required only one additional histogram in this example, which is shown also in the test run output

(4) original and additional scatter plots
The output format for the original and additional histograms is identical to that in section 4.3.2.

(5) Additional histograms and scatter plots from an external histogram package
If an external histogram package is used, their output is printed at this step.

(6) Frequency distribution of trials
A histogram of number of events versus number of trials to get an event is obtained.

If the frequency distribution has a sharp peak at the first bin, the generation efficiency is very high. On the contrary, if it is very broad and the number of unsuccessful trials (overflow bin) is large, the generation efficiency is very low. In the latter case, the grid sizes of subregions were not well optimized at the grid defining step of BASES. To improve the efficiency, it is recommended to submit BASES from the beginning with a larger number of NTRIAL or ITMX1, or with the smaller value of ACC1.

Acknowledgements

The primitive version of these packages has been developed at DESY, Hamburg by the author. We thank the F22 group and the computer center at DESY for their hospitality. We also thank Prof. Y. Shimizu at KEK for his valuable comments. Most of the routines in the appendices were given by him and Mr. J. Fujimoto. Finally, we wish to thank Professors H. Yoshiki, S. Iwata, T. Nozaki and Y. Oyanagi for reading the manuscript and useful discussion.

Appendix A. Example of JCL and MAIN program for BASES on FACOM machine at KEK

```
// JOB-ID JOB CLASS = S
// EXEC FORT7CLG,
//        PARM.FORT = 'OPT(3), S, GOSTMT, NONUM',
//        PARM.LKED = 'NOMAP, NOLIST'
// FORT.SYSIN   DD DSN = User-ID. NUPAIR.FORT(MAINB), DISP = SHR
//              DD = DSN = User-ID. NUPAIR.FORT(USERIN), DISP = SHR
//              DD DSN = User-ID. NUPAIR.FORT(USROUT), DISP = SHR
//              DD DSN = User-ID. NUPAIR.FORT(FUNC), DISP = SHR
// LKED.SYSLIB DD
//              DD DSN = User-ID. BASES.LOAD,DISP = SHR
// GO.SYSIN     DD *
   1, 1              Current loop count, Max. loop count
   5                 Print Flag
   0                 Input Flag
   1.0               CPU time limit in minutes
// *
// GO.FT23F001 DD DSN = User-ID.@BASES.DATA, DISP = (NEW,CATLG),
// SPACE = (TRK,(10,2),RLSE)
```

If the data set @BASES.DATA exists already, the last DD statement should be replaced by the following one:

```
//GO.FT23F001 DD DSN = User-ID.@BASES.DATA,DISP = SHR
```

List of main program MAINB:

```
      EXTERNAL FUNC

      CALL BSMAIN (FUNC)

      STOP
      END
```

Appendix B. Example of the USERIN subprogram

```
      SUBROUTINE USERIN

      IMPLICIT REAL * 8 (A-H, O-Z)
      COMMON/BASE1/ XL(10),XU(10),NDIM,NOCUB,NTRIAL
      COMMON/BASE2/ ACC1,ACC2,ITMX1,ITMX2

      COMMON/KINEM/ W,EM,ZM,ZGAM,CZ,CV,CA,FACTOR,XK, COSTH
      DATA PI/ 3.1415926D0/, ALP/137.036D0/
      DATA GEVNB/ 0.38927D6/, GENER/ 3.0/

      W      =105.0
      EM     =0.511E-3
      ZM     =90.0
      WM     =78.97
      ZGAM   =2.6
      ALPHA=1./ALP
      RAD    =PI/180.0
        SQ   =SQRT(ZM * * 2-WM * * 2)
        CZ   =ZM * * 2/(2. * WM * SQ)
        CA   =0.5 * CZ
        CV   =2. * CA * (-0.5 + 2. * SQ * * 2/ZM * * 2)
      FACTOR = GENER * CZ * * 2 * ALPHA * * 3/12. * GEVNB

      Initialize BASES parameters

      COSMIN  = 15.0
      COSMAX  = 180.-COSMIN
      XKMIN   = 1./W
      XL(1)   = XKMIN
      XU(1)   = 0.5 * W
      XL(2)   = DCOS(COSMAX * RAD)
      XU(2)   = DCOS(COSMIN * RAD)
      NDIM    = 2
      NTRIAL  = 5
      ITMX2   = 50
      ACC2    = 0.05
```

```
C Initialize Histograms

      CALL XHINIT (1, XL(1),XU(1),40, 'PHOTON ENERGY')
      CALL XHINIT (2, XL(2),XU(2),50, 'COS(THETA) of PHOTON')
      CALL DHINIT (1, XL(1),XU(1),50, XL(2),XU(2),50,
                   'PLOT FOR PHOTON ENERGY - COS(THETA) OFPHOTON')

      RETURN
      END
```

Appendix C. Example of the user function program

```
REAL FUNCTION FUNC * 8(X)

IMPLICIT REAL * 8(A-H, O-Z)
DIMENSION X(10)
COMMON/KINEM/W,EM,ZM,ZGAM,
CZ,CV,CA,FACTOR,XK,COSTH

      REZ(S)  = S-ZM * * 2
      Z2(S)   = (REZ(S)) * * 2 + (ZM * ZGAM) * * 2
      FUNC    = 0.
      XK      = X(1)
      S       = W * W
      S1      = W * (W-2. * XK)
      E       = W/2.
      PP      = SQRT(E * * 2-EM * * 2) COSTH = X(2)
      D1      = XK * (E + PP * COSTH)
      D2      = XK * (E-PP * COSTH)

C.....MATRIX ELEMENT SQUARE STARTS
      ANS1 = (S * * 2 * CA * * 2 + S * * 2 * CV * * 2 - 2. * S * W * XK * CA * * 2 - 2. * S *
     · W * XK * CV * * 2 - 3. * S * CA * * 2 * EM * * 2 - 3. * S * CV * * 2 * EM * * 2 + 2. *
     · W * XK * CA * * 2 * EM * * 2 + 2. * W * XK * CV * * 2 * EM * * 2)/D1 + S1 * (-S *
     · * 2 * CA * * 2 - S * * 2 * CV * * 2 + 10. * S * CA * * 2 * EM * * 2 - 2. * S * CV * * 2
     · * EM * * 2 - 16. * CA * * 2 * EM * * 4 + 8. * CV * * 2 * EM * * 4)/(2. * D1 *
     · D2) + (-S * * 3 * CA * * 2 - S * * 3 * CV * * 2 + 2. * S * * 2 + W * XK * CA * * 2 + 2.
     · * S * * 2 * W * XK * CV * * 2 + 2. * S * * 2 + CA * * 2 * EM * * 2 + 2. * S * * 2 * CV
     · * * 2 * EM * * 2 - 4. * S * W * XK * CA * * 2 * EM * * 2 - 4. * S * W * XK * CV * * 2
     · * D1 * D2) - (D2 * S1 * EM * * 2) * (CA * * 2 + CV * * 2)/D1 * * 2 + D2 * EM * * 2 *
     · (-s * CA * * 2 - S * CV * * 2 + 2. * W * XK * A * * 2 + 2. * W * XK * CV * * 2)/D1 *
     · * 2 + S1 * EM * * 2 * (S * CA * * 2 + S * CV * * 2 - 8. * CA * * 2 * EM * * 2 + 4. * CV
     · * * 2 + EM * * 2)/(2. * D1 * * 2) + S * EM * * 2 * (S * CA * * 2 + S * CV * * 2 - 2. * W
     · * XK * CA * * 2 - 2. * W * XK * CV * * 2)/(2. * D1 * * 2)
      TAU = -(D1 * S1) * (CA * * 2 + CV * * 2)/D2 + D1 * (-S * CA * * 2 - S * CV * * 2 + 2.
     · * W * XK * CA * * 2 + 2. * W * XK * CV * * 2 + 2. * CA * * 2 * EM * * 2 + 2. * CV * *
     · 2 * EM * * 2)/D2 - (D1 * S1 * EM * * 2) * (CA * * 2 + CV * * 2)/D2 * * 2 + D1 * EM *
```

```
     · * 2 * (-S * CA * * 2 - S * CV * * 2 + 2. * W * XK * CA * * 2 + 2. * W * XK * CV * *
     · 2)/D2 * * 2 + 4. * EM * * 2 * (CA * * 2 + CV * * 2) + S1 * (S * CA * * 2 + S * CV * *
     · 2 - 3. * CA * * 2 * EM * * 2 - 3. * CV * * 2 * EM * * 2)/D2 + (S * * 2 * CA * * 2 + S *
     · * 2 * CV * * 2 - 2. * S * W * XK * CA * * 2 - 2. * S * W * XK * CV * * 2 - 3. * S * CA
     · * * 2 * EM * * 2 - 3. * S * CV * * 2 * EM * * 2 + 2. * W * XK * CA * * 2 * EM * *
     · 2 + 2. * W * XK * CV * * 2 * EM * * 2)/D2 + S1 * EM * * 2 * (S * CA * * 2 + S * CV *
     · * 2 - 8. * Ca * * 2 * EM * * 2 + 4. * CV * * 2 * EM * * 2)/(2. * D2 * * 2) + S * EM * * 2
     · * (S * CA * * 2 + S * CV * * 2 - 2. * W * XK * CA * * 2 - 2. * W * XK * CV * * 2)/(2. *
     · D2 * * 2) - (D2 * S1) * (CA * * 2 + CV * * 2)/D1 + D2 * (-S * CA * * 2 - S * CV * *
     · 2 + 2. * W * XK * CA * * 2 + 2. * W * XK * CV * * 2 + 2. * CA * * 2 * EM * * 2 + 2. *
     · CV * * 2 * EM * * 2)/D1 + S1 * (S * CA * * 2 + S * CV * * 2 - 3. * CA * * 2 * EM * *
     · 2 - 3. * CV * * 2 * EM * * 2)/D1 + ANS1

      DSDX = -FACTOR * TAU * XK/(E * Z2(S1))
      FUNC = DSDX/E
      CALL XHFILL(1,XK,DSDX)
      CALL XHFILL(2,COSTH,DSDX)
      CALL DHFILL(1,XK,COSTH,DSDX)

        RETURN
        END
```

Appendix D. Example of JCL and MAIN program for SPRING on FACOM machine at KEK

```
// JOB-ID JOB CLASS = S
// EXEC FORT7CLG,
//        PARM.FORT = 'OPT(3),S,GOSTMT,NONUM',
//        PARM.LKED = 'NOMAP,NOLIST'
//        FORT.SYSIN   DD DSN = User-ID. NUPAIR.FORT(MAINS),DISP = SHR
//                     DD DSN = User-ID. NUPAIR.FORT(USERIN),DISP = SHR
//                     DD DSN = User-ID. NUPAIR.FORT(FUNC),DISP = SHR
//                     DD DSN = User-ID. NUPAIR.FORT(SPINIT),DISP = SHR
//                     DD DSN = User-ID. NUPAIR.FORT(SPEVNT,DISP = SHR
//                     DD DSN = User-ID. NUPAIR.FORT(SPTERM),DISP = SHR
//        LKED.SYSLIB DD
//                     DD DSN = User-ID. BASES.LOAD,DISP = SHR
//        GO.SYSIN     DD *
        100000  Number of generating events
        1.0     CPU time limit in minutes
//       *
//        GO.FT23F001 DD DSN = User-ID. @BASES.DATA,DISP = SHR
//        GO.FT20F001 DD DSN = User-ID.@NUPAIR.EVENT.DATA, DISP = SHR
```

List of main program MAINS:

```
      EXTERNAL FUNC
      MXTRY = 50
```

```
CALL SPMAIN(FUNC, MXTRY)

  STOP
  END
```

Appendix E. Example of SPINIT and SPEVNT

```
  SUBROUTINE SPINIT

  CALL XHINIT(5, 0.0, 4.0D1, 40,
    'PHOTON TRANSVERSE ENERGY')
  RETURN
  END

  SUBROUTINE SPEVNT

  IMPLICIT REAL * 8 (A-H, O-Z)
  REAL * 4 VECTOR(4)
  COMMON/KINEM/W,EM,ZM,ZGAM,CZ, CV,CA,FACTOR,XK,COSTH
  DATA TWOPI, ONE/ 6.2831853D0, 1.0/

  PHI       =TWOPI * DRN(IDUM)
  SINPH     =SIN(PHI)
  COSPH     =COS(PHI)
  VXY       =XK * SQRT(1.0-COSTH * COSTH)
  VECTOR(1) =VXY * COSPH
  VECTOR(2) =VXY * SINPH
  VECTOR(3) =XK * COSTH
  VECTOR(4) =XK

  WRITE(20) VECTOR

  CALL XHFILL(5, VXY, ONE)

  RETURN
  END
```

Appendix F. Descriptions of program components and labelled commons of BASES and SPRING

SUBROUTINE BSTIME (TIME,IFLG)

Purpose: Interface routine to get used CPU time from FORTRAN Library routine CLOCK, etc.

Comment: We use FORTRAN library routine CLOCK for this purpose. This routine should be replaced by a relevant routine in other system. (Used by BASES and SPRING.)

TIME=: Used CPU time in second.

IFLG=: Input flag. If IFLG=0, initialize clock routine, otherwise return the used time.

SUBROUTINE BSUSRI

Purpose: To check the BASES parameters given by USERIN.

Comment: In this routine USERIN is called. (Used by BASES.)

SUBROUTINE BSREAD
Purpose: To read parameters for BASES and temporary results of the previous job from unit number 23.
Comment: This routine is called after USERIN when IFLAG > 0. (Used by BASES and SPRING.)

SUBROUTINE BASES (FXN)
Purpose: To integrate the user function FXN and to control the integration.
FXN =: User function name.

SUBROUTINE BSINIT
Purpose: To initialize BASES program only for the case of IFLAG = 0.
Comment: In this routine, the numbers of hypercubes and small-hypercubes are defined and the uniform grid for them is assumed. (Used by BASES.)

FUNCTION DRN (IQ)
Purpose: To get double precision uniform random number.
IQ =: Dummy argument.

SUBROUTINE DRNSET (X1, X2)
Purpose: To set seeds of random number.
X1, X2 =: Seeds of random number.

SUBROUTINE DRNSAV (X1, X2)
Purpose: To get current seeds of random number.
X1, X2 =: Seeds of random number.

SUBROUTINE BSGDEF
Purpose: To adjust the grid for the next iteration according to the current distribution obtained by the previous iteration.
Comment: This routine is called only in the grid defining step. (Used by BASES.)

SUBROUTINE BSWRIT
Purpose: To write parameters for BASES and temporary results onto a file of unit number 23.
Comment: This routine is always called before terminating the job. (Used by BASES.)

SUBROUTINE BHINIT
Purpose: To reset the histogram and the plot.
Comment: This routine is used only for use of the loop option.

SUBROUTINE BHRSET
Purpose: To reset data of the histogram and plot.
Comment: This routine is used only once before starting the accumulation step. (Used by BASES.)

SUBROUTINE BHUPDT (WGHT)
Purpose: To update the contents of histograms and plots.
Comment: The routines XHFILL and DHFILL identify the bin to be updated. This routine updates the content of the bin with the weight of F * WGHT. (Used by BASES.)
WGHT =: Inverse of the probability for selecting the sub-hypercube.

SUBROUTINE BHPLOT
Purpose: Interface routine to print histograms and scatter plots, where routines XHPLOT and DHPLOT are called. (Used by BASES.)

SUBROUTINE SPRING (FXN, MXTRY, NTRY)
Purpose: To generate the event (SPRING).
FXN =: Name of the user function.
MXTRY =: Maximum number of trials defined by main routine.
NTRY =: Number of trials for generating the current event.

SUBROUTINE SHRSET
Purpose: To reset the data of histograms and plots
Comment: This routine is called only once before starting the event generation (SPRING).

SUBROUTINE SHCLER
Purpose: To cancel the update of histograms and plots in case where the trial was not accepted as an event.
Comment: This routine is called prior to call SHUPDT, when the trial is rejected. (Used by SPRING).

SUBROUTINE SHUPDT
Purpose: To update the contents of histograms and plots.

Comment: The routines XHFLL and DHFILL identify the bin to be updated. This routine update the content of the bin with unit weight (SPRING).

SUBROUTINE SHPLOT
Purpose: To print histograms and scatter plots defined by XHINIT and DHINIT, respectively.
Comment: For a histogram, there are two types of outputs, one is for the original histogram and the other is for the additional one. For the original histogram, the generated events is shown with "○" mark on the histogram of BASES result to make comparison easy. For the additional histograms, the generated results are shown both in linear and logarithmic scales. (SPRING).

SUBROUTINE XHPLOT (IFLG, ID)
Purpose: To print histograms defined by XHINIT. (Used by BASES SPRING.)
IFLG=: Flag which indicates whether it is called by BASES or SPRING.
ID=: Histogram ID number.

SUBROUTINE DHPLOT
Purpose: To print scatter plots defined by DHINIT. (Used by BASES SPRING.)

COMMON/BASED/SETIME,UTIME, IFLAG,NPRINT
SETIME=: CPU time set by user, converted into second unit.
UTIME=: Used CPU time in second.
IFLAG=: Job input flag described in section 4.1.
NPRINT=: Print flag.

COMMON/BASE1/XL(10),XU(10),NDIM, NOCUB,NTRIAL
Description of parameters is given in section 4.2.

COMMON/BASE2/ACC1,ACC2,ITMX1, ITMX2
Description of parameters is given in section 4.2.

COMMON/BASE3/SI,SI2,SWGT,SCHI, SCALLS,ATACC,NSU,IT
SI,SI2,SWGT,SCHI=: Parameters by which cumulative estimate of the integration is calculated.
SCALLS=: Total number of sampling points.
ATACC=: Accuracy of the previous iteration.
NSU=: Number of iterations whose accuracies are within 5% of ATACC.
IT=: Current iteration number.

COMMON/BASE4/XI(50, 10),DX(10), XD(17000),DXP(17000),ND,NG,NPG,MA(10)
XI(50, 10)=: Sampling intervals of sub-regions along each variable axis.
DX(10)=: Differences between the lower and upper limits of the integration for each variable.
DXD(i)=: Storage area for average probabilities.
DXP(i)=: Storage area for maximum probabilities.
ND=: Number of subregions.
NG=: Number of regions.
MA(10)=: Parameters by which mapping betwen hypercubes and addresses of storage area DXD(i) and DXP(i) is done.

COMMON/BASE5/TIME(100),EFF(100), WRONG(100),RESLT(100),ACST(100), TRSLT(100),TSTD(100),PCNT(100)
Labelled common /BASE5/ is for saving the convergence behavior from the beginning of the step either grid defining or accumulation.
TIME(i)=: Used CPU time up to the ith iteration.
EFF(i)=: Efficiency (=number of points used for estimate/total sampling points) for ith iteration in percentage.
WRONG(i)=: Percentage of getting negative integrand function.
RESLT(i)=: Cumulative result of the integration at the ith iteration.
ACST(i)=: Accuracy of the cumulative result.
TRSLT(i)=: Result only from the ith iteration.
TSTD(i)=: Accuracy of TRSLT(i).
PCNT(i)=: Accuracy of the cumulative result in percentage.

COMMON/BASE6/D(50, 10)
D(50, 10)=: Distribution of FUNC * *2, pro-

jected along each variable axis, by which new subregion sizes are determined in BSGDEF.

COMMON/RANDOM/X1,X2
X1,X2 =: Random number seeds.

COMMON/BSRSLT/AVGI,SD,CHI2A,STIME, ITF
Labelled common /BSRSLT/ is prepared for giving the result to user's routine USROUT. This routine is called at the end of the accumulation step only if the print flag is set equal to zero.
AVGI =: Final result of the integration.
SD =: Standard deviation.
CHI2A =: Chi-square per iteration.
STIME =: Used CPU time.
ITF =: Number of iterations performed.

COMMON/LOOP0/LOOPC
Labelled common /LOOP0/ is prepared for giving the current loop number to the routine USERIN.
LOOPC =: Current loop number.

COMMON /LOOP1/ LOOP,MXLOOP
Labelled common /LOOP1/ is for saving the current and the maximum loop numbers in disk file. The reason why we have two commons /LOOP0/ and /LOOP1/ for the loop count is to protect MXLOOP from accidental change by user.
LOOP =: Current loop number.
MXLOOP =: Maximum loop number.

CHARACTER * 64 TEXT,TITL
COMMON /PLOT0/ TEXT(10),TITL(10)
TEXT(i) =: Title of the ith histogram.
TITL(i) =: Title of the ith scatter plot.

COMMON /PLOT1/ NHIST,
MAPP(10),IFBASE(10),MHIST,MAPD(10)
NHIST =: Number of histograms defined.
MAPP(i) =: Correspondence table between serial number i and histogram ID number, i.e., ID = MAPP(i).
IFBASE(i) =: Flag which indicates whether the ith histogram is defined for BASES or not. If it is defined in BASES, a special histogram is obtained for comparison between BASES and SPRING at the end of SPRING.
MHIST =: Number of scatter plots defined.
MAPD(i) =: Correspondence table between serial number i and scatter plot ID number, i.e., ID = MAPD(i).

COMMON /PLOT2/ XLMAX(10),
XLMIN(10),NOBIN(10),DLS(10)
XLMAX(i) =: Upper limit for the ith histogram.
XLMIN(i) =: Lower limit for the ith histogram.
NOBIN(i) =: Number of bins for the ith histogram.
DLS(i) =: Size of one bin of the ith histogram.

COMMON /PLOT3/ NPNT(10),
NXPNT(52, 10),XLS(52, 10)
NPNT(i) =: Number of total sampling points for the ith histogram.
NXPNT(j, i) =: Number of sampling points for the jth bin for the ith histogram.
XLS(j, i) =: Distribution of the ith histogram.

COMMON /PLOT4/ XDMAX(10),XDMIN(10),
MXBIN(10),DXS(10),YDMAX(10),YDMIN(10),
MYBIN(10),DYS(10)
XDMAX(i) =: Upper limit of X for the ith scatter plot.
XDMMIN(i) =: Lower limit of X for the ith scatter plot.
YDMAX(i) =: Upper limit of Y for the ith scatter plot.
YDMIN(i) =: Lower limit of Y for the ith scatter plot.
MXBIN(i) =: Number of bins of X for the ith scatter plot.
MYBIN(i) =: Number of bins of Y for the ith scatter plot.
DXS(i) =: Size of X bin of the ith scatter plot.
DYS(i) =: Size of Y bin of the ith scatter plot.

COMMON /PLOT5/ MPNT(10),
XYDST(50, 50, 10)
MPNT(i) =: Number of sampling points for the ith scatter plot.
XYDST(k, j, i) =: Distribution of the ith scatter plots.

COMMON /PLOT6/ NOX(10),NTAG(5, 10),
XFX(5, 10),NOD(10),MXTAG(5, 10),MYTAG-
(5, 10),DFX(5, 10)

NOX(i) =: Flag which indicates whether the ith histogram is to be filled. If NOX(i) is zero, no XHFILL was called. If NOX(i) is positive and less than 6, XHFILL was called NOX(i) times.

NTAG(j, i) =: The bin number, which is to be updated by the weight XFX(j, i), of the ith histogram.

XFX(j, i) =: The weight of the jth calls of XHFILL for the ith histogram.

NOD(i) =: Flag which indicates whether the ith plot is to be filled. If NOD(i) is zero, no DHFILL was called. If NOD(i) is positive and less than 6, DHFILL was called NOD(i) times.

MXTAG(j, i) =: The bin number of X, which is to be updated by the weight DFX(j, i), of the ith plot.

MYTAG(j, i) =: The bin number of Y, which is to be updated by the weight DFX(j, i), of the ith plot.

DFX(j, i) =: The weight of the jth calls of DHFILL for the ith histogram.

References

[1] G.P. Lepage, J. Comput. Phys. 27 (1978) 192; VEGAS Adaptive Multi-dimensional Integration Program, CLNS-80/447 (March 1980).

[2] G. Barbiellini, B. Richter and J.L. Siegrist, Phys. Lett. 106B (1981) 414.

TEST RUN OUTPUT

IT	Time	Eff.	R_Neg	RESULT	ST-DEV	T-RES	T-ACC	Acc	Maximum of Dist = 1.00
1	1.77	100.0	0.00	0.453494D-01	0.146D-03	0.453D-01	0.322	0.322	I**************** I
2	1.98	100.0	0.00	0.453203D-01	0.101D-03	0.453D-01	0.311	0.224	I********** I
3	2.20	100.0	0.00	0.452977D-01	0.825D-04	0.453D-01	0.314	0.182	I********* I
4	2.41	100.0	0.00	0.452917D-01	0.721D-04	0.453D-01	0.327	0.159	I******* I
5	2.63	100.0	0.00	0.453253D-01	0.652D-04	0.455D-01	0.336	0.144	I******* I
6	2.84	100.0	0.00	0.453072D-01	0.592D-04	0.452D-01	0.313	0.131	I****** I
7	3.06	100.0	0.00	0.453101D-01	0.551D-04	0.453D-01	0.333	0.122	I****** I
8	3.27	100.0	0.00	0.453191D-01	0.517D-04	0.454D-01	0.331	0.114	I***** I
9	3.49	100.0	0.00	0.452962D-01	0.490D-04	0.451D-01	0.336	0.108	I***** I
10	3.70	100.0	0.00	0.452855D-01	0.462D-04	0.452D-01	0.309	0.102	I***** I
11	3.92	100.0	0.00	0.452947D-01	0.442D-04	0.454D-01	0.335	0.098	I**** I
12	4.13	100.0	0.00	0.452778D-01	0.417D-04	0.451D-01	0.280	0.092	I**** I
13	4.35	100.0	0.00	0.452750D-01	0.402D-04	0.452D-01	0.334	0.089	I**** I
14	4.56	100.0	0.00	0.452560D-01	0.385D-04	0.451D-01	0.291	0.085	I**** I
15	4.78	100.0	0.00	0.452528D-01	0.371D-04	0.452D-01	0.311	0.082	I**** I
16	4.99	100.0	0.00	0.452488D-01	0.359D-04	0.452D-01	0.310	0.079	I*** I
17	5.21	100.0	0.00	0.452537D-01	0.349D-04	0.453D-01	0.330	0.077	I*** I
18	5.42	100.0	0.00	0.452502D-01	0.340D-04	0.452D-01	0.346	0.075	I*** I
19	5.64	100.0	0.00	0.452616D-01	0.333D-04	0.455D-01	0.344	0.073	I*** I
20	5.85	100.0	0.00	0.452713D-01	0.325D-04	0.455D-01	0.335	0.072	I*** I
21	6.07	100.0	0.00	0.452793D-01	0.318D-04	0.455D-01	0.346	0.070	I*** I
22	6.28	100.0	0.00	0.452763D-01	0.309D-04	0.452D-01	0.294	0.068	I*** I
23	6.50	100.0	0.00	0.452681D-01	0.301D-04	0.451D-01	0.284	0.066	I*** I
24	6.71	100.0	0.00	0.452682D-01	0.294D-04	0.453D-01	0.301	0.065	I*** I
25	6.93	100.0	0.00	0.452553D-01	0.286D-04	0.450D-01	0.282	0.063	I*** I
26	7.14	100.0	0.00	0.452506D-01	0.280D-04	0.452D-01	0.289	0.062	I*** I
27	7.36	100.0	0.00	0.452527D-01	0.275D-04	0.453D-01	0.331	0.061	I*** I
28	7.57	100.0	0.00	0.452545D-01	0.270D-04	0.453D-01	0.322	0.060	I** I
29	7.79	100.0	0.00	0.452550D-01	0.266D-04	0.453D-01	0.322	0.059	I** I
30	8.00	100.0	0.00	0.452561D-01	0.262D-04	0.453D-01	0.340	0.058	I** I
31	8.22	100.0	0.00	0.452591D-01	0.257D-04	0.453D-01	0.314	0.057	I** I
32	8.43	100.0	0.00	0.452666D-01	0.254D-04	0.455D-01	0.329	0.056	I** I
33	8.65	100.0	0.00	0.452679D-01	0.250D-04	0.453D-01	0.311	0.055	I** I
34	8.86	100.0	0.00	0.452726D-01	0.246D-04	0.454D-01	0.319	0.054	I** I
35	9.07	100.0	0.00	0.452775D-01	0.243D-04	0.455D-01	0.335	0.054	I** I
36	9.29	100.0	0.00	0.452759D-01	0.239D-04	0.452D-01	0.320	0.053	I** I
37	9.51	100.0	0.00	0.452774D-01	0.236D-04	0.453D-01	0.301	0.052	I** I
38	9.72	100.0	0.00	0.452756D-01	0.233D-04	0.452D-01	0.317	0.051	I** I
39	9.93	100.0	0.00	0.452764D-01	0.230D-04	0.453D-01	0.345	0.051	I** I
40	10.15	100.0	0.00	0.452792D-01	0.228D-04	0.454D-01	0.330	0.050	I** I
41	10.36	100.0	0.00	0.452813D-01	0.225D-04	0.454D-01	0.320	0.050	I** I

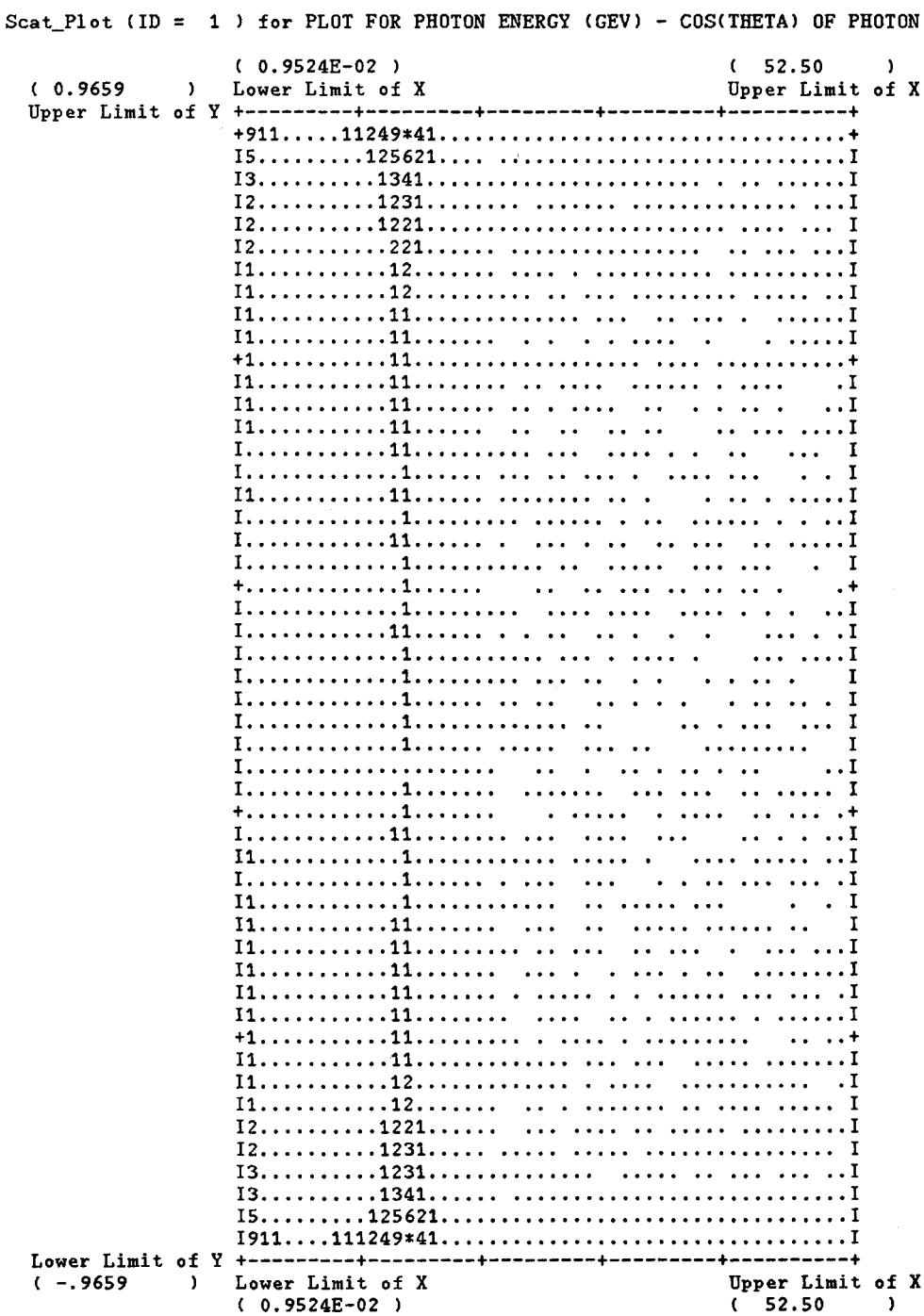
Histogram (ID = 1) for PHOTON ENERGY (GEV)

Linear Scale indicated by "*"

Lower value	d(Sigma)/dx	No_Sampling	0.000E+00	0.100	0.200	0.300	0.400	0.500
underflow	0.0000E+00	0						
0.9524E-02	0.3205	23034	**********	0000000000000000000000000000				
1.322	0.5420E-01	3855	********	00000000000000000000000000000000000000000000000000000000000000				
2.634	0.3784E-01	2588	*******	00000000000000000000000000000000000000000000000000000000000000				
3.946	0.3356E-01	2387	******	00000000000000000000000000000000000000000000000000000000000000				
5.259	0.3217E-01	2284	******	00000000000000000000000000000000000000000000000000000000000000				
6.571	0.3699E-01	2620	******	00000000000000000000000000000000000000000000000000000000000000				
7.883	0.4513E-01	3202	*******	00000000000000000000000000000000000000000000000000000000000000				
9.195	0.6210E-01	4315	*********	00000000000000000000000000000000000000000000000000000000000000				
10.51	0.1064	7324	***************	00000000000000000000000000000000000000000000000000000000000000				
11.82	0.2415	16752	**********************************	00000000000000000000000000000000000000000000000000				
13.13	0.4730	32790	**********	0000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000				
14.44	0.2314	15916	*******************************	00000000000000000000000000000000000000000000000000				
15.76	0.7006E-01	4785	**********	00000000000000000000000000000000000000000000000000				
17.07	0.2800E-01	1756	****	00000000000000000000000000000000000000000000000000				
18.38	0.1440E-01	994	**	00000000000000000000000000000000000000000000000000				
19.69	0.7602E-02	940	**	00000000000000000000000000000000000000000000000000				
21.01	0.4239E-02	115	*	00000000000000000000000000000000000000000000000000				
22.32	0.2866E-02	99	*	00000000000000000000000000000000000000000000000000				
23.63	0.2061E-02	107	*	00000000000000000000000000000000000000000000000000				
24.94	0.1602E-02	120	*	00000000000000000000000000000000000000000000000000				
26.25	0.1273E-02	135	*	00000000000000000000000000000000000000000000000000				
27.57	0.6278E-03	91	*	00000000000000000000000000000000000000000000000000				
28.88	0.6363E-03	124	*	00000000000000000000000000000000000000000000000000				
30.19	0.4301E-03	110	*	00000000000000000000000000000000000000000000000000				
31.50	0.3044E-03	105	*	00000000000000000000000000000000000000000000000000				
32.82	0.1936E-03	89	*	00000000000000000000000000000000000000000000000000				
34.13	0.1683E-03	100	*	00000000000000000000000000000000000000000000000000				
35.44	0.1345E-03	103	*	00000000000000000000000000000000000000000000000000				
36.75	0.1002E-03	99	*	00000000000000000000000000000000000000000000000000				
38.07	0.8289E-04	105	*	00000000000000000000000000000000000000000000000000				
39.38	0.6821E-04	113	*	00000000000000000000000000000000000000000000000000				
40.69	0.4737E-04	103	*	00000000000000000000000000000000000000000000000000				
42.00	0.3121E-04	87	*	00000000000000000000000000000000000000000000000000				
43.31	0.3051E-04	116	*	00000000000000000000000000000000000000000000000000				
44.63	0.2328E-04	120	*	00000000000000000000000000000000000000000000000000				
45.94	0.1395E-04	97	*	00000000000000000000000000000000000000000000000000				
47.25	0.1088E-04	111	*	00000000000000000000000000000000000000000000000000				
48.56	0.6270E-05	105	*	00000000000000000000000000000000000000000000000000				
49.88	0.4104E-05	126	*	00000000000000000000000000000000000000000000000000				
51.19	0.8678E-06	103	*	00000000000000000000000000000000000000000000000000				
overflow	0.0000E+00	0						

Lower value	d(Sigma)/dx	No_Sampling	0.100E-06	0.100E-04	0.100E-02	0.100

Logarithmic Scale indicated by "0"


Scat_Plot (ID = 1) for PLOT FOR PHOTON ENERGY (GEV) - COS(THETA) OF PHOTON
(0.9524E-02)
(52.50)
(0.9659)
Lower Limit of X
Upper Limit of X
Upper Limit of Y
Lower Limit of Y
(-.9659)
Lower Limit of X
Upper Limit of X
(0.9524E-02)
(52.50)

Original Histogram (ID = 1) for PHOTON ENERGY (GEV)

Total number of events = 100000 Orig. Dist. in Log Scale, indicated by "*"

Lower value	d(Sigma)/dx	Generated	0.100E-06	0.100E-04	0.100E-02	0.100
underflow	0.0000E+00	0				
0.9524E-02	0.3205	17742				
1.322	0.5420E-01	3007				
2.634	0.3784E-01	2188				
3.946	0.3356E-01	1858				
5.259	0.3217E-01	1790				
6.571	0.3699E-01	2046				
7.883	0.4513E-01	2421				
9.195	0.6210E-01	3390				
10.51	0.1064	5858				
11.82	0.2415	13281				
13.13	0.4730	26132				
14.44	0.2314	12727				
15.76	0.7006E-01	3944				
17.07	0.2800E-01	1540				
18.38	0.1440E-01	774				
19.69	0.7602E-02	418				
21.01	0.4239E-02	274				
22.32	0.2866E-02	177				
23.63	0.2061E-02	117				
24.94	0.1602E-02	82				
26.25	0.1273E-02	55				
27.57	0.6278E-03	48				
28.88	0.6363E-03	34				
30.19	0.4301E-03	16				
31.50	0.3044E-03	24				
32.82	0.1936E-03	15				
34.13	0.1683E-03	11				
35.44	0.1345E-03	7				
36.75	0.1002E-03	5				
38.07	0.8289E-04	5				
39.38	0.6821E-04	4				
40.69	0.4737E-04	1				
42.00	0.3121E-04	3				
43.31	0.3051E-04	0				
44.63	0.2328E-04	3				
45.94	0.1395E-04	0				
47.25	0.1088E-04	2				
48.56	0.6270E-05	0				
49.88	0.4104E-05	1				
51.19	0.8678E-06	0				
overflow	0.0000E+00	0				

Lower value d(Sigma)/dx Generated Generated Events, indicated by "0" (Arbitrary unit in Log)

Additional Histogram (ID = 5) for PHOTON TRANSVERSE ENERGY (GEV)

Total number of events = 100000 Number of Events in Linear Scale, indicated by "*"

Lower value	Log (dN/dx)	dN/dx	0.000E+00 0.500E+04 0.100E+05 0.150E+05 0.200E+05
underflow	0.0000E+00	0	
0.0000E+00	0.1888E+05	18880	I**********0000000000000000
1.000	3441.	3441	I**********0000000000000000000000000000000000000000000000000000000000000000
2.000	3122.	3122	I*********0000000000000000000000000000000000000000000000000000000000000000
3.000	6400.	6400	I*******************0000000000000000000000000000000000000000000000000000000000
4.000	9178.	9178	I*****************************0000000000000000000000000000000000000000000000000000
5.000	7863.	7863	I************************0000000000000000000000000000000000000000000000000000000
6.000	6752.	6752	I*********************0000000000000000000000000000000000000000000000000000000000
7.000	6173.	6173	I*******************0000000000000000000000000000000000000000000000000000000000
8.000	5769.	5769	I******************0000000000000000000000000000000000000000000000000000000000
9.000	5506.	5506	I*****************0000000000000000000000000000000000000000000000000000000000
10.00	5304.	5304	I*****************0000000000000000000000000000000000000000000000000000000000
11.00	5531.	5531	I*****************0000000000000000000000000000000000000000000000000000000000
12.00	5477.	5477	I*****************0000000000000000000000000000000000000000000000000000000000
13.00	5222.	5222	I*****************0000000000000000000000000000000000000000000000000000000000
14.00	2783.	2783	I*********0000000000000000000000000000000000000000000000000000000000000000
15.00	1178.	1178	I****0000000000000000000000000000000000000000000000000000000000
16.00	539.0	539	I**00000000000000000000000000000000000000000000000000000000
17.00	314.0	314	I**0000000000000000000000000000000000000000000000000000
18.00	171.0	171	I*000000000000000000000000000000000000000000
19.00	104.0	104	I*0000000000000000000000000000000000000
20.00	58.00	58	I*00000000000000000000000000000000
21.00	67.00	67	I*0000000000000000000000000000000000
22.00	45.00	45	I*000000000000000000000000000000
23.00	27.00	27	I*0000000000000000000000000
24.00	29.00	29	I*00000000000000000000000000
25.00	13.00	13	I*0000000000000000000
26.00	15.00	15	I*00000000000000000000
27.00	7.000	7	I*000000000000000
28.00	6.000	6	I*0000000000000
29.00	8.000	8	I*00000000000000000
30.00	1.000	1	I0
31.00	4.000	4	I*0000000000
32.00	4.000	4	I*0000000000
33.00	3.000	3	I*00000000
34.00	1.000	1	I0
35.00	0.0000E+00	0	I
36.00	1.000	1	I0
37.00	2.000	2	I*00000
38.00	1.000	1	I0
39.00	1.000	1	I0
overflow	0.0000E+00	0	I

Lower value Log (dN/dx) dN/dx 1.00 10.0 100. 0.100E+04 0.100E+05

Number of Events in Logarithmic Scale, indicated by "0"

********** Number of trials to get an event **********

Total number of events = 100000 Number of Events in Linear Scale, indicated by "*"

Lower value	Log (dN/dx)	dN/dx	0.000E+00	0.200E+05	0.400E+05	0.600E+05	0.800E+05
underflow	0.0000E+00	0					
1.000	0.8024E+05	80242	**********				
2.000	0.1370E+05	13696	**********				
3.000	3416.	3416	**********				
4.000	1057.	1057	**********				
5.000	561.0	561	**********				
6.000	298.0	298	**********				
7.000	185.0	185	**********				
8.000	122.0	122	**********				
9.000	86.00	86	**********				
10.00	87.00	87	**********				
11.00	48.00	48	**********				
12.00	53.00	53	**********				
13.00	30.00	30	**********				
14.00	23.00	23	**********				
15.00	21.00	21	**********				
16.00	16.00	16	**********				
17.00	14.00	14	**********				
18.00	6.000	6	**********				
19.00	11.00	11	**********				
20.00	7.000	7	**********				
21.00	7.000	7	**********				
22.00	4.000	4	**********				
23.00	1.000	1	O				
24.00	3.000	3	**********				
25.00	2.000	2	**********				
26.00	1.000	1	O				
27.00	0.0000E+00	0					
28.00	0.0000E+00	0					
29.00	0.0000E+00	0					
30.00	1.000	1	O				
31.00	1.000	1	O				
32.00	0.0000E+00	0					
33.00	0.0000E+00	0					
34.00	0.0000E+00	0					
35.00	0.0000E+00	0					
36.00	0.0000E+00	0					
37.00	1.000	1	O				
38.00	0.0000E+00	0					
39.00	0.0000E+00	0					
40.00	0.0000E+00	0					
41.00	0.0000E+00	0					
42.00	0.0000E+00	0					
43.00	0.0000E+00	0					
44.00	0.0000E+00	0					
45.00	0.0000E+00	0					
46.00	0.0000E+00	0					
47.00	0.0000E+00	0					
48.00	0.0000E+00	0					
49.00	0.0000E+00	0					
50.00	0.0000E+00	0					
overflow	0.0000E+00	0					

Lower value	Log (dN/dx)	dN/dx	1.00	10.0	100.	0.100E+04	0.100E+05

Number of Events in Logarithmic Scale, indicated by "O"