CGTM No. 195
April 1979

A User's Guide for SAGE

A General System for Monte Carlo Event Generation with Preferred Phase Space Density Distributions

by

Roger B. Chaffee

Stanford Linear Accelerator Center
Computation Group Technical Memo No. 195
April 1, 1979

Table of Contents

This writeup is dated January 18, 1980

I. Introduction

This memo is a condensation or outright copy of the information in CGTM No. 145 [Ref. 8]. The section on subroutine GENIUS has been added, and the descriptions have been elaborated for some of the more Baroque calling sequences of the other routines.

Users of earlier versions of SAGE should note that the common blocks /INIT/, /MDEL/, and /SWITCH/ have been replaced by the single block /SAGECM/, in an attempt to avoid conflicts. I hope this will not affect many users. INITL0 has been changed to have no arguments. The old call can be used with the IBM compilers, so this will not be a problem.

SAGE is a set of Fortran subroutines that generate Monte Carlo events in Lorentz-invariant phase space. These events may be used to simulate particle physics experiments or evaluate phase-space integrals.

In addition to generating events that correspond to a uniform population of phase space, SAGE can generate events with other phase-space density distributions. The additional phase-space density distributions available in SAGE include Breit-Wigner-@-, $\exp[-a|t|]$, and $1/(t-a^2)^2$ four-momentum transfer squared distributions, and general angular distributions. SAGE has standard options for generating events that correspond to 1)

-@- The Breit-Wigner function used in SAGE is

$$BW(m) = 1/[(m-E)^2/(W/2)^2 + 1]$$

Here m is the variable invariant mass and E and W are the parameters of the function, giving the central value and the full width at half maximum, respectively.

a uniform density in Lorentz-invariant phase space, 2) limited transverse momenta of final-state particles (peripheral phase-space), 3) limited successive four-momentum transfers squared in multiperipheral chains (multiperipheral phase-space). The weights of the resulting events can be modified according to any desired generation probability or detection efficiency factors to adjust them to the user's particular problem.

SAGE is initialized by subroutine INITL (Section II). Four-vectors for an event are generated by a call to GOGEN, GENIS, GENIUS, GODEL, or RY/GO (Section III), and these routines may be called in any combination for more complex events. The weight for an event is obtained by a call to WT (Section IV), which also resets SAGE for the next event.

SAGE can generate the four-vectors of the final-state particles in any Lorentz frame specified by the user. It can be used to calculate total and differential cross-sections predicted by dynamic models. It can simulate experiments with any distribution of initial-state center-of-mass energy, and it can be used to generate events according to several different frequency distributions for the same problem.

Input to SAGE consists of the kinematic parameters of the initial state (center-of-mass energy or beam momentum, beam and target mass), specification of the Lorentz frame in which the four-vectors of the final state particles are to be expressed, and the parameters that characterize the phase-space density distribution of the generated events. The output of SAGE consists primarily of the four-vectors for the final-state

particles in the specified Lorentz frame. Other output quantities may be available in the common block /SAGECM/, depending on the options selected.

The user can use these output quantities to calculate values for matrix elements predicted by dynamic models, for the purpose of simulating experiments or calculating phase-space integrals. SAGE may be used in conjunction with data summarizing programs (for example SUMX [1], KIOWA [2], and HPAK [9]) in order to histogram, scatter plot, or otherwise summarize the Monte-Carlo data. The output of SAGE can also be used as input to data-reduction programs for the study of experimental biases.

II. Initialization

Before any events are generated, or whenever the initial-state parameters are changed, SAGE must be initialized. This may be done with one of the following Fortran statements-@-:

```
1)   CALL INITL (BEAM,TARGET,ECMO,DUMMY,P)
2)   CALL INITL (BEAM,TARGET,ECMO,PBEAM,SYSTEM)
3)   CALL INITL (BEAM,TARGET,ECMI,3HEND)
4)   CALL INITL0
```

Input Quantities

BEAM	REAL	scaler	Mass of beam particle
TARGET	REAL	scaler	Mass of Target particle
ECMI	REAL	scaler	Center-of-mass energy
PBEAM	REAL	scaler	Beam momentum in SYSTEM frame
P	REAL	(4)	Initial state four-vector.
SYSTEM	Hollerith		2HCM (center-of-mass system) or 3HLAB (target at rest)
DUMMY	REAL	scaler	Argument required, but not used by SAGE.

Output Quantities

ECMO	REAL	scaler	Center-of-mass energy

In addition, there is a common block which will contain the beam and target four-momenta:

```
COMMON /SAGECM/ TRNSFR,COSINE,BM(4),TG(4),ISW
```

BM is the four-vector of the beam and TG that of the target, expressed in the same Lorentz frame as the initial state four-vector. If SYSTEM rather than P is used, the beam is taken to be in the +z direction. This common storage is filled by SAGE in the call to INITL (but not INITL0), and is used by the event generation subroutines.

-@- Some compilers will not allow truncated calling sequences--the number of arguments in a call must equal the number of arguments in the subroutine declaration. In any case, arguments beyond the 3HEND are not processed by SAGE, so any value could be used. For instance,

```
CALL INITL (BEAM,TARGET,ECM,3HEND,0.)
```

would be fine. For compilers which also check the type and dimension of the argument, such as WATFIV, alternate calling sequences are probably more trouble than they are worth.

1) CALL INITL(BEAM,TARGET,ECMO,DUMMY,P)

P defines the Lorentz frame in which the final-state four-vectors are to be generated. For example, P = (0,0,0,ECM) will cause them to be generated in the overall center-of-mass. P = (0,0,PBEAM,ELAB+TARGET) will cause them to be generated in the laboratory system (the frame in which the target is at rest). The transverse components of P would be non-zero if the final state results from the decay of a particle with four-momentum P, or if the beam and the target are not collinear, as in a colliding-beam experiment.

ECMO is calculated by INITL from P.

2) CALL INITL(BEAM,TARGET,ECMO,PBEAM,SYSTEM)

Since the center-of-mass and the laboratory systems are the ones most frequently used, special calls may be used for them. If SYSTEM specifies one of these values, a four-vector is not needed to specify the overall frame. In this case, the magnitude of the beam momentum (PBEAM) must be given.

System	SYSTEM	Equivalent P
Center-of-Mass	2HCM	(0,0,0,ECM)
Laboratory	3HLAB	(0,0,PBEAM,ELAB+TARGET)

ECMO is calculated by INITL from the input parameters.

3) CALL INITL(BEAM,TARGET,ECMI,3HEND)

In this case, the frame is assumed to be the center-of-mass system, and the c.m. energy must be supplied by the calling program.

4) CALL INITL0

This call may be used only if there is no angular dependence in the final state density distribution. Obviously, this is true in a limited set of problems. In particular, INITL0 may not be used with GENIS, GENIUS, GODEL, or any RY/GO sequence which has DEL, POL, or COS generation. The beam and target four-vectors in common block /SAGECM/ are not filled by INITL0.

III. Event Generation

A Monte-Carlo event is obtained from SAGE by one or more calls to SAGE subroutines. The number of calls for each event, and the subroutines called, are governed by the phase-space density distribution desired. SAGE has four subroutines that can generate an entire event with one call. These routines generate events corresponding to a uniform phase-space density distribution (GOGEN), limited transverse momenta of the final-state particles (GENIS & GENIUS), and limited successive four-momentum transfers squared in multiperipheral chains (GODEL). More general phase-space density distributions are possible. These require more than one subroutine call to SAGE for each event. Each of these options is discussed in turn below.

A. GOGEN (Uniform Phase-Space Distribution)

GOGEN uses the technique of [3] to general final-state particle four-vectors corresponding to a uniform phase-space density distribution. GOGEN may be used to generate all or some of the final state four-vectors. GOGEN performs a series of calls to RY and GO. It generates a multiparticle system A = (A1,A2,...,An) by factoring the n-body decay vertex into a special set of two-body decay vertices. The first vertex recoils (A1,A2,...,A(n-1)) from An. The second recoils (A1,...,A(n-2)) from A(n-1), and so on, until the last recoils A1 from A2. A1 may be a multiparticle system, but A2,...,An must all be single particles.

This multiparticle system is generated by the single subroutine call

```
     CALL GOGEN(U,P,N,S,U1,P1)
or   CALL GOGEN(U,SYSTEM,N,S,U1,P1)
```

Input variables

U	REAL scaler	Invariant mass of A system
P	REAL (4)	Four-vector of A system.
SYSTEM	Hollerith	2HCM: Return 4-vectors in A rest frame. 3HLAB: Return them in rest frame of the target. (This works only if this is the only call to a SAGE event generator, or the first of a combination.)
N	INTEGER (2)	N(1) = number of particles comprising A system N(2) = number of particles comprising A1 system
S	REAL (N)	Rest masses S(1) = sum of masses of A1 system S(2) = mass of A2 ... S(N) = mass of AN

Output Quantities

U1	REAL	(N)	Invariant masses of multiparticle recoiling systems U1(1) = mass of A1 U1(2) = mass of A1+A2 ... U1(N-1) = mass A1+...+A(N-1) U1(N) not used.
P1	REAL	(4,N)	Final-state four-vectors. P1(I,1) 4-vec of A1 ... P1(I,N) 4-vec of AN

An alternative call is permitted, if only the invariant masses U1(1),...,U1(N-1) are required. In this case the call is

```
CALL GOGEN(U,DUMMY,N,S,U1,3HEND)
```

B. GENIS (Peripheral Phase Space)

GENIS uses the technique of [4] to generate final state four-vectors with a distribution of transverse momenta corresponding to $dN/d(Pt^2) = \exp[-Pt^2/R^2)]$. Note that GENIUS is much more efficient at performing the same task. The Pt are the components of the particles' momentum transverse to the beam direction, and $R^2 = \langle Pt^2 \rangle$ at infinite energy. As with GOGEN, GENIS may be used to generate all or part of the final state. Events of this type are obtained by the single subroutine call

```
      CALL GENIS(U,P,N,S,P1,R)
or    CALL GENIS(U,SYSTEM,N,S,P1,R)
```

Subroutine INITL must be used to set up for GENIUS. Do not use subroutine INITL0.

Input Quantities

U	REAL	scaler	Invariant mass of all N particles
P	REAL	(4)	Four-momentum of all N particles
SYSTEM	Hollerith		2HCM (Beam in +z direction) (3HLAB is not allowed)
N	INT	scaler	number of particles to generate
S	REAL	(N)	Rest masses S(1) = mass of first ... S(N) = mass of n-th
R	REAL	(4)	parameter (see text)

Output Quantities

P1	REAL	(4,N)	Output four-vectors P1(I,1) = 4-vec of A1 ... P1(I,N) = 4-vec of AN

The parameter R determines the distribution of the transverse momenta of the generated particles. The distribution $dN/d(Pt^2)$ corresponds to multiplying phase space by the matrix

element $|M|^2 = \exp[-SUM(Pt^2/r^2)]$, with $r^2 = [\ N/(N-1)\ ]\ R^2$. The factor N/(N-1) is due to a delta-function for momentum conservation: the average Pt is constrained by forcing the sum of the transverse momenta to be zero. The value of R should be adjusted to obtain the best generation efficiency for the user's particular matrix element squared. Experimentally it is found that R is about 0.4 GeV, so that is probably a good starting value. Procedures for adjusting generation parameters in order to maximize generating efficiencies in SAGE are discussed in [8].

D. GENIUS (Peripheral Phase Space)

GENIUS uses the technique of [7] to generate final state four-vectors with transverse momenta taken from one of several distributions. The calling sequence is similar to that of GENIS, with the addition of parameters to describe the distribution of the transverse momenta. These may have a Gaussian (GAUS), Exponential (EXP), or Power (POWR) distribution. Even with gaussian distribution, which is the form generated by subroutine GENIS, the distribution of weights is more uniform than for those generated by GENIS, and event generation is much more efficient.

$$\text{GAUS:}\quad |M|^2 = \exp\left[-\sum_{i=1}^{N} Pt(i)^2/r^2\right]$$

$$\text{EXP:}\quad |M|^2 = \exp\left[-\sum_{i=1}^{N} |Pt(i)|/r\right]$$

$$\text{POWR:}\quad |M|^2 = \prod_{i=1}^{N} [m^2/(Pt(i)^2+m^2)]^{**r}$$

N is the number of particles to be generated. As with GOGEN, GENIS may be used to generate all or part of the final state. Events of this type are obtained by the single subroutine call

```
      CALL GENIUS(U,P,N,S,P1,R)
or    CALL GENIUS(U,SYSTEM,N,S,P1,R)
```

Subroutine INITL must be used to set up for GENIUS. Do not use subroutine INITL0.

Input Quantities

U	REAL	scaler	Invariant mass of all N particles
P	REAL	(4)	Four-momentum of all N particles
SYSTEM	Hollerith		2HCM (2HLAB is not allowed)
N	INT	scaler	number of particles to generate
S	REAL	(N)	Rest masses S(1) = mass of first ... S(N) = mass of n-th
R	REAL	(4)	R(1)=r R(2)=Keyword 4HGAUS, 4HEXP , or 4HPOWR R(3)=Blowup factor (Use 0.0). R(4)=m (EXP generation only)

Output Quantities

P1	REAL	(4,N)	Output four-vectors P1(I,1) = 4-vec of A1 ... P1(I,N) = 4-vec of AN

Event generation for the GAUS distribution produces the same weighted distributions as are produced by subroutine GENIS. The EXP and POWR distributions have no analog in other SAGE routines.

The value of R should be adjusted to obtain the best generation efficiency for the user's particular matrix element squared. Experimentally it is found for GAUS distributions that R is about 0.4 GeV, so that is probably a good starting value. Procedures for adjusting generation parameters in order to maximize generating efficiencies in SAGE are discussed in [8].

D. GODEL (Multiperipheral Phase Space)

GODEL uses the technique of [5] to generate final state particle four-vectors with a phase-space density corresponding to

$$\text{DEL:} \quad |M|^2 = \prod_{i=1}^{N-1} e^{-a(i)t(i)}$$

$$\text{POL:} \quad |M|^2 = \prod_{i=1}^{N-1} \frac{1}{[t(i)-a(i)^2]^2}$$

and is discussed in detail in that reference. The t(i) are the successive four-momentum transfers squared in a multiperipheral chain. The parameters a(i) characterize the distribution of the t(i) for the generated particles. As with GOGEN and GENIS, GODEL may be used to generate all or part of the final state. GODEL factors the N particle system into two-body decay vertices in exactly the same manner as GOGEN. The notation used below to describe the chain of vertices is the same as in section III-A (page 10).

Events with "DEL" or "POL" distributions are obtained by the subroutine call

```
     CALL GODEL(U,P,N,S,P1,PARM,FLAG)
  or CALL GODEL(U,SYSTEM,N,S,P1,PARM,FLAG)
```

Input Quantities

U	REAL	scaler	Invariant mass of A system
P	REAL	(4)	Four-vector of A system
SYSTEM	Hollerith		2HCM or 3HLAB
NP	INT	(2)	NP(1) = number of particles comprising A system NP(2) = number of particles comprising A1 system Note that N = NP(1)-NP(2)+1 and that for one particle in the A1 system, N = NP(1) and NP(2) = 1
S	REAL	(N)	Rest masses S(1) = sum of masses of A1 system S(2) = mass of A2 ... S(N) = mass of AN
PARM	REAL	(4)	Parameters that characterize the phase-space density distribution (see [5]) PARM(1) = b0 (Eq. 20) PARM(2) = E0 (Eq. 20) PARM(3) = d1 (Eq. 25) PARM(4) = d' (Eq. 25)
FLAG	Hollerith		4HBDEL, 4HTDEL, 4HBPOL, or 4HTPOL (Table below)

Output Quantities

P1	REAL	(4,N)	Output four-vectors P1(I,1) = 4-vec of A1 ... P1(I,N) = 4-vec of AN

FLAG Table

FLAG	Meaning
4HBDEL	"DEL" distribution, t defined with respect to (w.r.t.) beam
4HTDEL	"DEL" distribution, t defined w.r.t target
4HBPOL	"POL" distribution, t defined w.r.t beam
4HTPOL	"POL" distribution, t defined w.r.t target

E. Generalized Phase Space Density Distributions

SAGE generates an n-particle final state by factoring it into (n-1) two-body systems, and generating each system as a two-body decay A --> A1+A2. [Ref. 3] Here A is the parent system comprised of N ≤ n particles and A1 and A2 are its daughters, comprised of M1 and M2 particles, respectively. (Note that M1≥1, M2≥1, and N=M1+M2.) The way in which the final state is factored determines which invariant masses and four-momentum transfers squared can be given Breit-Wigner or exp(at) or $1/(t-a^2)^2$ density distributions. A particular factoring of an n-particle final state yields a particular set of n-2 invariant masses and n-1 momentum transfers squared that may (but need not) be given these density distributions. The user specifies the phase-space density distribution of the generated events both by the way in which he factors the final state into two-body decays and by the way in which each decay is generated. The choice as to how to factor the final state is decided by the user by coding a SAGE "event type". The event type consists of a series of calls to SAGE subroutines. Each call generates a two-body decay vertex. The user connects these vertices to form the final state [3].

Consider a decay vertex A --> A1 + A2. The invariant masses of the A1 and A2 systems may be generated according to phase space or Breit-Wigner density. (See footnote, p 3) The decay angular distribution of A1 or A2 in the A rest frame may be generated flat or according to exp(at) or $1/(t-a^2)^2$, where t is the four-momentum transfer squared from either the beam or the target. This decay angular distribution may also be generated

according to an arbitrary angular distribution, measured in the A rest frame, supplied by the user, with either the beam, target, or helicity directions as the reference direction.

In order to generate a vertex the user must make <u>two</u> calls to SAGE:

```
CALL RY(S1,S2,U,P,E,D)
CALL GO(P1,P2,U1,U2)
```

or

```
CALL RY(S1,S2,U,SYSTEM,E,D)
CALL GO(P1,P2,U1,U2)
```

The first call provides the necessary input and the second returns the output from the generation.

RY Input Quantities

S1	REAL	(2)	A1 mass description S1(1) = sum of rest masses of A1 system S1(2) = number of particles in A1 system
S2	REAL	(2)	A2 mass description S2(1) = sum of rest masses of A2 system S2(2) = number of particles in A2 system
U	REAL	scaler	Invariant mass of A system
P	REAL	(4)	Four-momentum of A system
SYSTEM	Hollerith		2HCM: Return 4-vectors in A rest frame. 3HLAB: return 4-vectors in rest frame of target, as defined by call to INITL.
E	REAL	(4)	Breit-Wigner parameters E(1) = Central A1 mass. = 0.0 if no B-W dependence E(2) = A1 width E(3) = Central A2 mass. = 0.0 if no B-W dependence E(4) = A2 width
D	REAL	(3)	Angular distribution description D(1) = Flag. (See table below) D(2) = 1. for generating A1 D(2) = 2. for generating A2. D(3) = parameter a. for exp[at] "DEL" or $1/(t-a^2)^2$ "POL" D(3) = cos(-) for cos(-) "COS" generation. For COS generation, - is measured in the rest frame of A.

FLAG Table

FLAG	Meaning
4HBDEL	"DEL" distribution, t defined with respect to (w.r.t.) beam
4HTDEL	"DEL" distribution, t defined w.r.t target
4HBPOL	"POL" distribution, t defined w.r.t beam
4HTPOL	"POL" distribution, t defined w.r.t target
4HBCOS	"COS" distribution, cosine defined w.r.t beam
4HTCOS	"COS" distribution, cosine defined w.r.t target
4HHCOS	"COS" distribution, w.r.t. helicity direction of A

Output Quantities

P1	REAL	(4,2)	Four-vectors for A1 P1(I,1) = 4-vector in given frame P1(I,2) = 4-vector in A rest frame
P2	REAL	(4,2)	Four-vectors for A2 P2(I,1) = 4-vector in given frame P2(I,2) = 4-vector in A rest frame
U1	REAL	scaler	Invariant mass of A1 system
U2	REAL	scaler	Invariant mass of A2 system

In addition, there is a common block which will contain the momentum transfer and cosine of the decay, whenever a vertex is generated according to an exp[at] or $1/(t-a^2)^2$ distribution.

```
COMMON /SAGECM/ TRNSFR,COSINE,BM(4),TG(4),ISW
```

3. Abbreviated Calls The calling sequences for RY and GO may be shortened for certain standard situations, using the Hollerith string 3HEND to signify the end of the argument list:

CALL RY(S1,S2,U,P,E,3HEND) implies D(I)=0., I=1 to 3

CALL RY(S1,S2,U,P,3HEND) implies E(I)=0., I=1 to 4, also

CALL RY(S1,S2,U,3HEND) implies SYSTEM=2HCM also

CALL R0(S1,S1,U) is the same as CALL RY(S1,S2,U,3HEND)

CALL GO(P1,P2,U1,3HEND) is allowed if S2(2)=1.

CALL GO(P1,P2,3HEND) is allowed if S1(2)=S2(2)=1.

In order to generate the invariant mass(es) U1 (and U2) at a vertex, the generator needs only S1, S2, and U. Thus, the N-2 invariant masses of an N-particle final state may be generated without generating any four-vectors. Alternatively, if only the four-vectors above a certain vertex are needed, then those below the vertex need not be generated. However, to make the weights

come out right, the masses below that vertex must be generated, whether used or not. Since most of the computational time in generating an event goes into constructing the four-vectors, a considerable saving can be realized by generating only the masses when the momenta are not needed. Note, however, that if a vertex decay angular distribution is not flat, then all the four-vectors above and including that vertex must be generated.

In order to generate only the masses at a vertex, the user must call G0 instead of GO, with the following calling sequence:

CALL G0(U1,U2) if both A1 and A2 are multiparticle systems, or, optionally,

CALL G0(U1,3HEND) if A2 is a single particle, or

CALL G0(3HEND) if both A1 and A2 are single particles.

If the vertex is to be generated flat in decay angles, then SYSTEM may be set to 2HCM in the call to RY, since the invariant masses are the same in any system.

4. Two-step Generation with RY and GO

In the common block

```
COMMON /SAGECM/ TRNSFR,COSINE,BM(4),TG(4),ISW
```

ISW is an optional switch which allows the generation of an event to be a two-step process.

ISW = 0 (or not set by user): Usual generation. This is the default.

ISW = 1: Generate only masses, even though calls to RY and GO indicate that 4-vectors are to be generated.

ISW = 2: Using the masses which were generated with ISW = 1, generate 4-vectors as indicated in the calls to RY and GO.

Thus, the user may execute a series of calls to RY and GO with ISW = 1 and generate only masses, then set ISW = 2 and re-execute the same calls to generate the 4-vectors.

For this mode of operation, it is necessary to store all of the masses from the ISW = 1 calls in separate variable locations in the calling routine. They must not be changed before the calls with ISW = 2.

The user may also generate the masses arbitrarily, independent of SAGE, and then by using only ISW = 2, have SAGE generate the 4-vectors corresponding to these masses. Also, if at any decay vertex, the user wishes to generate an invariant mass independent of SAGE, he may set S1(2) or S2(2) equal to 1.0, even though the invariant mass represents a system of more than one particle. In this case, SAGE will use the value of S1(1) or S2(1) as the input invariant mass for the system.

F. Generating According to a Decay Angular Distribution

As noted above, any vertex may be given an arbitrary decay angular distribution I(cos-). This is done by setting the D-array in the call to RY, for that vertex, as described in Section III-D (p. 18). D(1) and D(2) describe the coordinate system, and D(3) contains the cosine of the angle, cos-. This cosine is an input value supplied by the calling routine. It can be generated as follows:

$$\text{Let } F(\cos -) = \int_{E_{INT}}^{AL_R} I(\cos -')\, d(\cos -')$$

where I(cos-) is the desired decay angular distribution. For each event, generate a random number $0 = F(-1) \leq R \leq F(1)$, and set

$$F(\cos -) = R$$

Then cos - for the event is found by solving this equation for cos -, i.e.

$$\cos - = F^{-1}(R)$$

IV. Phase Space Weight

The events generated by SAGE are not distributed correctly in Lorentz-invariant phase space. A weight must be applied to each event to obtain the correct phase-space distribution.

After the entire event has been generated, the "phase-space" weight for the event is obtained by the call

```
CALL WT(W)
```

Output Quantities

W	REAL	(2)	Weights for the event W(1) = "phase-space" weight W(2) = W(1) x $\|M\|^2$

Calling WT also flags the end of an event for SAGE, so that the next call starts a new event. If an incomplete event is discarded, WT, INITL, or INITL0 must be called, even though the weight is not used, to initialize SAGE for the next event.

A complete discussion of phase-space weighting is given in [3.]. Only the details necessary for its use are given here.

W(2) contains the product of the Breit-Wigner, exp[at], $1/(t-a^2)^2$, I(cos-), and transverse momentum distributions applied at each vertex in the generation, and should be multiplied by any factors which represent a correction to these distributions. W(1) contains only the phase-space weight of a generated event, and should be multiplied by the entire matrix element squared. For many applications, it is sufficient to use the weight W(2) as calculated by SAGE, with corrections for the experimental detection efficiency. For more complex matrix elements, a common technique is to generate events by any of the SAGE routines, choosing the routine and parameters which give the best match to

the desired distribution. In this case, the weight for an event is W(1) times whatever $|M|^2$ is needed, and the event weight W(2) is not used at all.

V. Variable Center-of-Mass Energy

In order to more accurately simulate experimental conditions, the center-of-mass energy supplied to SAGE may vary from event to event. These energies can come from a frequency distribution P(E) determined by the experimental setup. Note that INITL must be called every time the c.m. energy is changed.

VI. Unweighted Events

SAGE returns a weight for each generated event, and the weight must be used in calculating distributions of quantities associated with these events. Some applications, however, require unweighted events with a given distribution. These can be produced by generating along with each weighted event a random number, x, in the interval (0,xmax), where xmax is an upper bound for the event weights. In order to produce unweighted events, those events are discarded for which the weight is less than x, and the retained events are given unit weight. These events are distributed with the given density.

The upper bound, xmax, need not be the least upper bound for the event weights, but the closer it is to the least upper bound, the more efficient the method becomes. This method always uses more SAGE processing time than using all the events with their original weights. However, if the Monte-Carlo events are

processed further after they are generated and the weight evaluated, then it may in fact be more economical to use unweighted events.

A method of approximating the least upper bound is to perform a search for the maximum event weight in the (3n-4) dimensional space of the randon numbers used to generate the event.

SAGE provides a method for performing this search. The approximate least upper bound is obtained by the call

```
CALL MAXWT(M,NRAND,NPRINT,WTMAX)
```

and by coding a function

```
FUNCTION RATE(M)
```

which will be called by MAXWT, and which must generate an entire event and return the event weight as the function value.

Input Quantities

M	INT	scaler	Not used by MAXWT, but passed to RATE for possible use there.
NPRINT	INT	scaler	NPRINT=0 suppresses all printing For NPRINT>0, every NPRINT-th search step is printed.

Output Quantities

WTMAX	REAL	scaler	Maximum event weight found.

As noted above, the search is performed in the space of the random numbers used to generate the events. The technique of the search assumes that the weight is a "nice" function in this space, which may or may not be the case. Events generated using RY/GO, GOGEN, and GODEL require NRAND=(3N-4) random numbers for each event, where N is the number of particles in the final state. If the calling program uses additional random numbers,

they should be obtained with the subroutine call

```
CALL RAND(R,NR)
```

where NR = Number of numbers desired.

R = Real array of length NR, which will contain them.

In this case, care must be taken in determining NRAND, which must be the total number of random numbers used by the called and calling routines.

The generation technique used in GENIS and GENIUS involves sampling and rejection in the space of random numbers, and different events may require different numbers of numbers. For this reason, MAXWT is not suitable for use with GENIS or GENIUS.

VII. Examples

In the following Fortran examples, lower case indicates numerical values to be supplied according to the particles and energies involved.

Generation with RY/GO

The following code will generate the final-state four-vectors P1, P2, and P3, and the weight W for the reaction Beam + Target --> A1 + A2 + A3, with uniform phase-space density.

```
      REAL S1(2),S2(2),S3(2),S12(2),W(2)
      REAL P12(4,2),P1(4,2),P2(4,2),P3(4,2)
      DATA X /3HEND/
      DATA S1 /m1,1.0/
      DATA S2 /m2,1.0/
      DATA S3 /m3,1.0/
      DATA ECM /ecm/
C                     INITIALIZATION (NEED EXECUTE ONLY ONCE)
      S12(1)=S1(1)+S2(1)
      S12(2)=S1(2)+S2(2)
      CALL INITL0
C                     EVENT GENERATION (ONCE PER EVENT)
C               GENERATE ECM --> S12 + S3
      CALL RY(S12,S3,ECM,2HCM,3HEND)
      CALL GO(P12,P3,U12,X)
C               AND S12 --> S1+S2
      CALL RY(S1,S2,U12,P12,3HEND)
      CALL GO(P1,P2,X)
C               GET THE WEIGHT
      CALL WT(W)
```

If only the A3 (or A1+A2) four-vector is needed, the event generation could be changed to

```
C                    EVENT GENERATION (ONCE PER EVENT)
C                GENERATE ECM --> S12 + S3
      CALL RY(S12,S3,ECM,2HCM,3HEND)
      CALL GO(P12,P3,U12,X)
C                AND S12 --> S1+S2
      CALL RO(S1,S2,U12)
      CALL GO(X)
C                GET THE WEIGHT
      CALL WT(W)
```

in order to save computational time.

If a cut is to be made on the energy of A3, then the event generation could be changed to

```
C                    INITIALIZATION (NEED EXECUTE ONLY ONCE)
      S12(1)=S1(1)+S2(1)
      S12(2)=S1(2)+S2(2)
   10 CALL INITLO
C                    EVENT GENERATION (ONCE PER EVENT)
C                GENERATE ECM --> S12 + S3
      CALL RY(S12,S3,ECM,2HCM,3HEND)
      CALL GO(P12,P3,U12,X)
C                MAKE THE CUT
      IF (P3(4).GT.2.0) GO TO 10
C                GENERATE S12 --> S1+S2
      CALL RY(S1,S2,U12,2HCM,3HEND)
      CALL GO(X)
C                GET THE WEIGHT
      CALL WT(W)
```

in order to save computational time.

An Example with More Complex Distributions

Here is a section of code which generates events for the reaction pi + p --> p + pi + pizero. The pions resonate at the rho mass, the rho is produced with an exp[-8|t|] distribution, and the rho decays with a $\cos^2(-)$ angular distribution. The calls to HIST are included to indicate how one could call an event-accumulating routine. Depending on whether you use KIOWA, as in this example, HPAK, SUMX, or some other program, you will have to make different calls.

```
      REAL S23(2),S1(2),S2(2),S3(2),P(4),P23(4,2),P1(4,2)
      REAL W(2),E(4),D(3),E3(4),D3(3),P2(4,2),P3(4,2)
C
      DATA  PLAB/13.7/
C                                PARAMETERS FOR REACTION 1
      DATA E/0.77,0.15,0.,0./,D/4HBDEL,1.0,8.0/
      DATA S23/0.27455,2./, S1/0.93826,1./
C                                PARAMETERS FOR REACTION 2
      DATA E3/0.,0.,0.,0./,D3/4HBCOS,1.,0./
      DATA S2/0.13958,1./, S3/0.13497,1./
C                                SET UP
      EBM=SQRT(PLAB**2 +0.13958**2)
      CALL INITL(0.13958,0.93826,ECM,PLAB,4HLAB )
      DO 100 N=1,500
C                                REACTION 1 -- PI+P --> RHO + P
        CALL RY(S23,S1,ECM,4HLAB ,E,D)
        CALL GO(P23,P1,U23,U1)
C                                REACTION 2 -- RHO --> PIZERO + PI
        R=RAN9(DUMMY)*2.-1.
        D3(3) = ABS(R)**(1./3.)
        IF (R.LT.0) D3(3) = -D3(3)
        CALL RY(S2,S3,U23,P23,E3,D3)
        CALL GO(P2,P3,U2,U3)
        CALL WT(W)
C                                EVENT IS GENERATED. NOW BIN IT
C                            MOMENTUM TRANSFER
        TRAN=-(0.13958**2 + U23**2 - 2.*P23(4,1)*EBM
     X         + 2.*PLAB*P23(3,1))
        CALL HIST(4,TRAN,W(2))
C                                RHO MASS
        CALL HIST(1,U23,W(2))
C                                DELTA MASS
        DELTA=(P2(4)+P1(4))**2
        DO 80 I=1,3
 80     DELTA=DELTA-(P2(I)+P1(I))**2
        CALL HIST(2,SQRT(DELTA),W(2))
 100  CONTINUE
```

An Example with Colliding Beams.

The accumulation routines in this example (HDEF1, HCLR, HCUM1, and HOUT) are part of the HPAK [9] package.

```
C         E+E- --> D+ D
C           D+ --> K PI PI
C           D- --> K PI
      DIMENSION PD1(4,2), PD2(4,2), POUT(4,5)
      DIMENSION N(2), XMASS(3), DUMMY(3), S1(2), E(4), D(3), W(2)
      DATA XMD /1.9/, XMK /0.494/, XMPI /.140/, ECM /4.4/
      DATA E /4*0.0 /, D /4HBCOS,1.0,0.0/
C
      EBM = ECM*.5
      CALL INITL(.000511,.000511,ECM,3HEND)
      CALL HDEF1(1,'E*4',20,0.,.05,' XE @')
      CALL HCLR('ALL')
C
      DO 100 I=1,1000
C            E+E- --> D+D- WITH SIN**2 THETA DIST.
   10    COSINE=2.*RAN7(0)-1.
         IF ( RAN7(0) .GT. 1.-COSINE**2 ) GO TO 10
         S1(1) = XMD
         S1(2) = 1.0
         D(3) = COSINE
         CALL RY ( S1, S1, ECM, 2HCM, E, D )
         CALL GO ( PD1, PD2, DUMMY, DUMMY )
C                  D+ --> K PI PI
         N(1) = 3
         N(2) = 1
         XMASS(1) = XMK
         XMASS(2) = XMPI
         XMASS(3) = XMPI
         CALL GOGEN ( XMD, PD1, N, XMASS, DUMMY, POUT )
C                  D- --> K PI
         N(1) = 2
         CALL GOGEN ( XMD, PD2, N, XMASS, DUMMY, POUT(1,4) )
         CALL WT(W)
C
C              HISTOGRAM ENERGY OF K-S THAT ARE DETECTABLE
         PXYSQ = POUT(1,1)**2 + POUT(2,1)**2
         COSINE = POUT(3,1) / SQRT(PXYSQ+POUT(3,1)**2)
         IF ( ABS(COSINE).LT.0.7 .AND. PXYSQ.GT.0.01 )
     *     CALL HCUM1(1,POUT(4,1)/EBM,W(2))
         PXYSQ = POUT(1,4)**2 + POUT(2,4)**2
         COSINE = POUT(3,4) / SQRT(PXYSQ+POUT(3,4)**2)
         IF ( ABS(COSINE).LT.0.7 .AND. PXYSQ.GT.0.01 )
     *     CALL HCUM1(1,POUT(4,4)/EBM,W(2))
  100 CONTINUE
      CALL HOUT('ALL','PRINTER')
      STOP
      END
```

References

1. L. Champomier, SUMX - A Data Summarizing Program, Lawrence Radiation Laboratory Report No. UCRL 11222, April 1964 (unpublished).

2. J.H. Friedman and A.R. Rittenberg, KIOWA - A General Description, Lawrence Radiation Laboratory Group A Programming Note No. P-171, May 1968 (unpublished).

3. J.H. Friedman, Journal of Comp. Physics, 7, 1 (1971).

4. L. Van Hove, Nucl. Phys., B9, 331 (1969); W. Kittel, L. Van Hove,and W. Wojcki, CERN Report No. CERN/DPhII/Phys 70-8, 1970.

5. J.H. Friedman, G.R. Lynch, C.G. Risk, and T.A. Zang, Jr, An Efficient Monte Carlo Event Generation Method for Multiperipheral Models, Lawrence Radiation Laboratory Report No. UCRL 20141, November 1970.; published in Journal of Comp. Physics, 8, 1 (August 1971).

6. S. Derenzo, MINF68 - A General Minimizing Routine, Lawrence Radiation Laboratory Group A Programming Note No. P-190, 1969 (unpublished).

7. A.E. Brenner, D.C. Carey, R. Pordes, J.H. Friedman, Fermilab Computer Dept. Report PM-4.

   David C. Carey and Daniel Drijard, Journal of Comp. Physics, 28, 327 (September 1978).

8. J.H. Friedman, SAGE, Stanford Linear Accelerator Center Compuation Group Technical Memo No. 145, December 1972 (unpublished).

9. C.A. Logg, Adam M. Boyarski, A. James Cook, and R. Leslie Cottrell, 'DPAK' AND 'HPAK': A VERSATILE DISPLAY AND HISTOGRAMMING PACKAGE. SLAC-0196 (Jun 1976).

A. Running SAGE at SLAC

Load modules for the SAGE routines are in the dataset WYL.CG.PUB.SAGEMODS, and can be made available to the loader by a parameter in the EXEC statement of your JCL. For instance,

```
//    EXEC  FORTGCG,
//       LKEDLB3='WYL.CG.PUB.SAGEMODS'
```

The source is a Wylbur Edit format dataset in WYL.CG.PUB.SAGE. Although SAGE is distributed freely, maintenance is at best an informal process. If you make your own copy, you will cut yourself off from any bug fixes or other improvements.

B. Program Notes

There is no subroutine SAGE in SAGE.

Subroutines RY and GO were written by Jerry Friedman at L.B.L. in the late 1960's. GOGEN and GODEL were added later.

GENIS was developed at CERN (by Kittel, van Hove and Wojcki?). This version was obtained by Jerry Friedman, and has been considerably modified by myself and Susan Cooper.

GENIUS is my own adaptation of routines used in the program NVERTX, which was developed at Fermilab and sent to me in May 1977 by Dave Carey.

Most of the code in most of the routines in this version of SAGE has been modified, to the point where the original authors should in no way be held responsible for any errors. I am solely responsible for any problems which you may find. I take an active interest in maintaining and even developing SAGE, and I will investigate any errors you find or suggestions you wish to make. For the good of the entire particle-physics community, please do not fix a bug only in your own copy of SAGE, but let me know about it too.

RBC