GRACE manual

— *Automatic Generation of Tree Amplitudes in Standard Models* —

Version 1.0

MINAMI-TATEYA group

KEK Report 92-19
February 1993

GRACE manual

— Automatic Generation of Tree Amplitudes in Standard Models —

Version 1.0

T.Ishikawa[1], T.Kaneko[2], K.Kato[3], S.Kawabata[1],
Y.Shimizu[1] and H.Tanaka[4]

[1]*National Laboratory for High Energy Physics(KEK),*
Tsukuba, Ibaraki 305, Japan
[2]*Faculty of general education, Meiji-gakuin University,*
Totsuka, Yokohama 244, Japan
[3]*Department of physics, Kogakuin University,*
Tokyo 160, Japan
[4]*Faculty of general education, Rikkyo University,*
Tokyo 171, Japan

Acknowledgements

On the way to develop the program system GRACE we have received much encouragement and support from many people. We would like to thank our colleagues in TRISTAN theory group, particularly Y. Kurihara, T. Munehisa, N. Nakazawa and J. Fujimoto for discussions which were helpful for getting the system better. We are also grateful to express our sincere gratitude to Professors H. Sugawara, S. Iwata and J. Arafune for the encouragement. Discussions with colleagues in Nuclear Physics Institute of Moscow State University and LAPP, Laboratoire d'Annecy-le-Vieux de Physiques des Particules, with whom we have been collaborating on the automatic calculation of Feynman amplitudes during these years, were very fruitful and useful for this work.

We are indebted to companies Fujitsu limited, Intel Japan K.K., KASUMI Co. Ltd and SECOM Co. Ltd. for their kind supports and understanding our work. A part of the calculations was performed on FACOM M1800, AP1000, VP series, S series, HITAC S820, M880, 3050, HP 9000 and Intel iPSC/860.

This work was supported in part by Ministry of Education, Science and Culture, Japan under Grant-in-Aid for International Scientific Research Program No.03041087 and No.04044158.

Chapter 1

Introduction

1.1 What is the problem ?

During the last two decades, it has been established that the gauge principle governs the interactions between elementary particles. In electroweak theory, leptons and quarks are interacting through exchange of three kinds of gauge bosons, photon, Z^0 and $W^{\pm}$. The assumed gauge group is $SU(2)_L \times U(1)$ and the original gauge symmetry is broken by the non-zero vacuum expectation value of Higgs field. On the other hand strong interaction between quarks is described by color $SU(3)$ gauge group. All the experimental facts seem to support these theories at present. Though it is still an open question how these different kinds of forces are unified into more fundamental theory, it is now of no doubt that these theories contain some truths and will remain as effectively correct ones.

This success of gauge theories or standard models of elementary particles, implies that we have definite Lagrangians and thus we can, in principle, predict any process based on these Lagrangians in perturbation theory. When one wants to perform calculation in this way, however, one meets a technical difficulty due to the complexity of the interaction Lagrangian. This is particular to non-abelian gauge theory in which we have three- and four-point self-couplings of gauge bosons as well as interactions of unphysical particles such as Goldstone bosons or ghost particles in general covariant gauge fixing. Hence even in the lowest order of perturbation, that is, in tree level, one finds a number of diagrams for a given process when the number of final particles increases. For example, we have only 3 diagrams for $e^+e^- \to W^+W^-$, but when one photon is added, $e^+e^- \to W^+W^-\gamma$, then 18 diagrams appear even after omitting the tiny interaction between $e^{\pm}$ and scalar bosons(Higgs and Goldstone bosons). Addition of one another photon, $e^+e^- \to W^+W^-\gamma\gamma$, yields 138 diagrams. Further if one wants to make more realistic calculation around the threshold of $W^{\pm}$ pair production, taking into account the decay of $W^{\pm}$, say, $W^- \to e^-\bar{\nu}_e$ and $W^+ \to u\bar{d}$, then one has to consider 24 diagrams for $e^+e^- \to e^-\bar{\nu}_e u\bar{d}$ and 202 for $e^+e^- \to e^-\bar{\nu}_e u\bar{d}\gamma$. In unitary gauge, as only physical particles appear in the Lagrangian, the numbers of diagrams are less than those mentioned above.

One may think that it is enough to select several diagrams which dominate the cross section. Even if one can find such dominant diagrams, one has to respect the gauge invariance among this subset of diagrams. Usually number of diagrams in the gauge invariant subset is not so small. For example, for the process $e^+e^- \to \nu_e\bar{\nu}_e W^+W^-$, we have 60 diagrams in all. Among them 30 diagrams form one gauge invariant set and the rest does another one. Hence still we meet the same difficulty to handle with many diagrams. In addition, there remains a possibility that the experimental cuts imposed on the final particles renders the dominant diagrams to be less prominent and all diagrams give somehow the same order of magnitude to the cross section. If this is the case, one has to keep whole the diagrams in the calculation after all.

Through the numerous experiments done at e^+e^- colliders, we have learned that higher order corrections should be included when we want to compare theories with experimental data in detail. This implies that we have to calculate at least one-loop corrections to a given process. As an example, consider the process $e^+e^- \to W^+W^-\gamma$. To regularize the infrared divergence due to soft photon emission, we have to include loop diagrams for $e^+e^- \to W^+W^-$ beyond the tree level, which contain virtual photon exchange and remove the divergence when combined with real photon emission process. The requirement of gauge invariance among one-loop diagrams demands, in turn, inclusion of other one-loop diagrams with exchanges of Z^0, $W^\pm$ or other possible particles. Then it is clear that the total number of diagrams becomes very huge and it is almost impossible even to enumerate all diagrams. In many cases it seems out of ability of mankind. For simple W-pair production, in general covariant gauge, the number is around 200 diagrams in the same approximation stated above, but for $e^-\bar{\nu}_e u\bar{d}$ it amounts more than 3,700.

Facing to the difficulty described above, we cannot help to find some ways to get rid of. As a solution we can choose the following one: As diagrams are constructed based on a set of *definite rules*, Feynman rules, it is natural to develop a computer code which can generate all the diagrams to any process, once initial and final particles are given. It should be able not only to enumerate diagrams but also generate automatically relevant amplitudes to be evaluated on computers, in other words, create a FORTRAN source code ready for amplitude calculation. `GRACE` (Ref.[3]) is such a system that realizes this idea and help us to reduce the most tedious part of works.

1.2 What we can do with `GRACE` ?

Before introducing what `GRACE` system can provide, let us remind the standard way to calculate cross sections at the tree level. Usually it consists of the following several different steps:

1) Specify the process.

2) Choose appropriate models.

3) Fix the order of perturbation(at the tree level, this is unique).

4) Enumerate all possible diagrams.

5) Write down amplitudes.

6) Prepare the kinematics for final particles.

7) Integrate the amplitude squared in the phase space of final particles, including experimental cuts, if necessary.

8) Generate events so that the simulation of the process in a detector is available.

9) Check the results.

Among these steps the first three, 1), 2) and 3), are trivial matter. For the step 7) one can rely on well established programs which are designed to make integration of multi-dimensional variables. This is of no problem, except for CPU-time, once the kinematics, step 6), is written so that the estimate of the integral is reliable within required accuracy. The step 8) is related with the preceding step. The last step 9) could be done to compare the results with other calculations or with approximated one. Hence the most tedious steps are 4) and 5). `GRACE` is a system of program packages for this purpose, namely, it carries out these most tedious steps on computers to save our elaboration.

1.2.1 What `GRACE` provides us?

The present version of `GRACE` generates:

- All the diagrams for a given process up to *one-loop*, when the order of perturbation is fixed(covariant gauge).
- FORTRAN source code which contains helicity amplitude of the process in the *tree level*(covariant gauge).
- Default values of all physical constants, *except for the strong coupling constant.*
- Interface routines to the program package `CHANEL` (Ref.[4]), which contains subroutines designed to evaluate the amplitude.
- *No kinematics is generated.*
- Interface routines to the multi-dimensional integration package `BASES` (Ref.[6]).
- Interface routines to the event generation package `SPRING` (Ref.[6]).
- Test program for gauge invariance check of the generated amplitude.
- Any diagram and its amplitude can be omitted in the calculation by setting the appropriate flags off. In the integration step the unitary gauge is the default(see section 3.2.1).

What the user should do first is to tell `GRACE` the set of parameters which specifies the process considered. It should include

1) names of initial particles,
2) names of final particles,
3) order of perturbation in QED, electroweak and/or QCD.

in the given format explained later.

When a job is started with the data file containing these inputs, `GRACE` constructs all possible diagrams and creates an output file to draw all Feynman diagrams for the convenience of the user to look them by eyes. At the same time a set of FORTRAN subprograms is generated. These include those which are needed to calculate the amplitude with the help of `CHANEL`, to integrate over phase space by `BASES` and to generate events by `SPRING`.

After all the programs are successfully generated, the next tasks, which the user should do before integration, are

1) to prepare the kinematics,
2) to fill up some parameters in a few subroutines, such as the dimension of the integral,
3) to check the gauge invariance of the amplitude.

1.2.2 Structure of the system

In this subsection we show how the whole system of `GRACE` is constructed and how each step proceeds. The system consists of the following four subsystems, whose interrelation is depicted in figure 1.1.

(1) **Graph generation subsystem**
When initial and final states of the elementary process are given as the input as well as the orders of couplings, a complete set of Feynman graphs is generated according to the theoretical model defined in a model definition file. For the time being QED, Electroweak and QCD models in the tree and one-loop level are supported. The information of generated graphs is stored in a file as an output.

Reading the graph information from the file, the graph drawer displays the Feynman graphs on the screen under the X-Window system or prints them on a paper.

(2) **Source generation subsystem**
From the graph information produced by the first subsystem, a FORTRAN source code is generated in a form of program components suited for the numerical integration package `BASES`.

The source code is constructed based on our helicity amplitude formalism, which consists of many calling sequences of subprograms given in CHANEL and its interface routines.

In addition to these program components, the subsystem generates a main program, by which the gauge invariance of the generated amplitudes can be tested.

Fig. 1.1 Structure of GRACE system

(3) **Numerical integration subsystem**
Combining the generated source code together with the kinematics routines and the GRACE library, the numerical integration is performed by BASES to obtain the total cross section. For this, however, one has to prepare the kinematics routines, which are discussed in the next section. As the output of integration, the numerical value of total cross section, the convergency behavior of integration, one and two dimensional distributions of the cross section are given besides the probability information in a file, which is used in the event generation. Looking the convergency behavior carefully one can judge if the resultant value is reliable or not.

(4) **Event generation subsystem**
Using almost all the same subprograms in the integration, events with *weight one*

are generated by the event generation program SPRING. To achieve a high generation efficiency, it uses the probability information produced by BASES. Conceptually, SPRING samples a point in the integration volume according to the probability. If the probability information is a complete one, the sampled point is exactly corresponding to a generating event. Since, however, it is impossible to get a complete information numerically, the sampled point is tested whether it is accepted or not. If it is accepted then its numerical values of integration variables are passed to an user interface routine SPEVNT, where they are transformed into the four vectors of the event.

1.2.3 How to do with kinematics ?

In order to get the numerical value of cross section, we make integration in the phase space of final particles. As the integral is multi-dimensional, 4 for 3-body, 7 for 4-body and 10 for 5-body process(if the cylindrical symmetry is assumed around the initial beam axis), we usually use adaptive Monte Carlo integration packages. (In our system BASES is assumed.) We have to express all momenta (or equivalently invariants composed of them) of final particles by independent integration variables. Generally speaking, the integration routine feeds a set of random numbers in the space of given dimension. Let us denote these random numbers as

$$\mathtt{X(I)}, \mathtt{I} = 1, \cdots, \mathtt{NDIM},$$

and assume their values are normalized in, say, $[0, 1]$. (In BASES, the upper and lower bounds for X(I) can be arbitrary numbers.) Then we have to translate these variables into four-momentum of final particle, say J-th particle, P(1,J), P(2,J), P(3,J), P(4,J) of total N particles (in GRACE, P(4,J) is the energy),

$$\mathtt{X(I)} \Longrightarrow \mathtt{P(K,J)}. \quad \mathtt{K} = 1, \cdots, 4, \quad \mathtt{J} = 1, \cdots, \mathtt{N}$$

This is known as kinematics for the given process. This mapping is not always unique and in some cases a single value of X(I) may correspond to multi-value of particle momenta.

GRACE, unfortunately, does not give the kinematics in an automatic way. The reason is that the present popular integration packages, such as BASES or VEGAS, utilize a special algorithm to search for the singularities of the integrand. The matrix element squared, the integrand, becomes singular when the denominators of propagators of internal particles become very small compared with the typical energy of the process considered. This happens when a mass of an internal line is very small. As is well known, if a singularity is running along the diagonal in a plane of two integration variables, these programs cannot give reliable estimate of the integral, because they fail to catch the singularity at all. In order to get good convergence of the integration over many iterations, all the singularities must be parallel to the integration axes. This means that these peaks located in the space of kinematical variables, are mapped onto the line of constant value of some `X(I)`. In order to do this, we have to choose very carefully the transformation between random numbers and kinematical variables. The typical kinds of singularities we meet in real calculation are as follows;

- mass singularity
- infrared singularity
- t-channel photon exchange
- resonance formation(decay of heavy particles)

(Precise description of how to deal with these singularities will be found in section 2.6).

In some processes the number of independent variables is greater than that of singularities, and one can easily find a kinematics which is suitable to make them smooth. If this is not the case, however, one may not be able to find such good kinematics to avoid diagonal singularity even after much efforts. Hence it is quite difficult to give the general kinematics which is capable of dealing with all kinds of singularities at once, or a single set of transformations.

The drawback adherent to the present integration packages mentioned above made us to hesitate to generate a kinematics, because it must be applicable only to limited processes. If GRACE could generate such kinematics, someone might apply it to a cross section which is so singular that the integration package fails to catch any singularity. It returns an answer which looks like to converge well at the first sight, but is completely wrong. Hence we decided not to generate kinematics automatic way, but leave it to the user.

1.2.4 How to make preliminary check ?

Suppose we have a kinematics for the process to be considered. The first task we should do is to check the generated amplitude and confirm that it is in fact correct one. We have two methods for this check;

1) Gauge invariance check.
   This is done by changing the gauge parameters numerically for γ, Z^0, $W^\pm$ and *gluon* and examining if the value of the total amplitude remains the same within the double precision. The main program for this test is generated by the system. When quadruple precision is supported on user's computer, invariance check in this precision level is also possible. One should, however, notice that in some special cases the gauge invariance is trivially satisfied and this kind of check cannot be helpful(simplest case is such that only the vector or axial couplings to on-shell massless fermions appears in each diagram).

2) Lorentz invariance check.
   Since all the four-components of particle momenta are numerically given, it is

possible to look if the squared amplitude does not change by Lorentz transformation. For this one has to change the definition of frame inside of the kinematics routine written by the user.

These tests can prove correctness of both amplitudes and kinematics. If either of these latter two has some errors, both invariance checks must fail. Note that, however, these cannot be responsible for the correctness of the overall factor multiplied to the squared amplitude(powers of 2π, factor 2, Jacobian in the kinematics and so on).
If everything is O.K., then you can proceed to make phase space integration.

1.3 How to use this manual

This manual is composed of three kinds of objects, theoretical background for calculating the cross section of elementary process, usage and technical details of the GRACE system. Throughout this manual we take the tree level process $e^+e^- \to W^+W^-\gamma$ as an example, *including* the $e^\pm$-scalar boson interactions. The real FORTRAN source code for this process is attached in the relevant sections as well as the results of calculation, which might be a great help for understanding the practical use of the system.

Structure of manual

The purpose of **chapter 2** is to present the theoretical background of the system. After notations, Lagrangians and renormalization prescriptions are specified, how amplitudes and color factors are calculated is described. The Feynman graph generation is briefly discussed from the graph theoretical point of view for the completeness of this manual.
Only the kinematics part is to be prepared by the user, where the structure of singularities in the phase space should be taken into account. The possible singularities, to which the user may face, are also discussed in this chapter. Finally the algorithms of multi-dimensional integration package and event generation package are presented.

Chapter 3 is devoted to the function of GRACE system, where full description about input and output of each sub-system is specified. Specification of subprograms for kinematics is also given here. This part is independent of the computer system, on which GRACE system is implemented.

In **chapter 4** the usage of GRACE system on UNIX system and FACOM main frame computer is described. GRACE system is also supported on some parallel computers. Usage on the parallel system INTEL iPSC/860 is presented as an example.

A variant of GRACE system for vector computers is described in **chapter 5**. The difference in the input and output specification of the vector version from the scalar one is mentioned. As an example usage of the system on HITAC S820/80 is presented.

In **chapter 6**, detailed description of Feynman rules is given. These rules are given to the system through a model definition file. The format of this file is also shown.

Chapter 7 is devoted to describe subroutines in CHANEL library and interface programs between CHANEL and generated code by GRACE.

The interrelation among the contents of chapters is shown in figure 1.2(c).

Traveling guide of this manual

Those serious users, who want to know how GRACE system is constructed and works before use, are recommended to read whole manual form the first page to the last. This will be the best but tedious way to understand GRACE system.

If you find interest only in physics and use conventional computers, you can start from chapter 4 and skip chapter 2 except for sections 2.1, 2.6 and chapters 5, 6 and 7 as shown in figure 1.2(a). When you intend to use a vector computer, you are recommended to start from chapter 5 on reference to chapters 2 and 3 as in figure 1.2(b).

Limitations of the system

This system has several limitations which are summarized in **appendix C**. It is recommended to read this appendix before using the system.

How to obtain GRACE system

GRACE full system works on UNIX workstations (at least HP and SUN) and main frame computers (FACOM and HITAC). In addition to these computers, the numerical subsystems of GRACE are applicable to VAX VMS system, parallel computers (INTEL iPSC/860, FACOM AP1000 and NCUBE), and vector computers (HITAC S series, FACOM VP series, NEC SX and CRAY). The system requires FORTRAN77 and PASCAL compilers for installation. For drawing generated Feynman graphs, it requires Xlib or GKS graphic library.

The system is available when requested through e-mail to `grace@minami.kek.jp`. Other information or question is welcome to the same e-mail address.

Fig. 1.2 Interrelation of chapters

Chapter 2

Theoretical background

In this chapter we describe general theoretical bases and ingredients used in the GRACE system. It covers conventions, definition of cross section, models, helicity amplitude formalism, calculation of color factor, method of graph generation, kinematics and the method of numerical integration. As the system automatically generates helicity amplitude for any tree process in the framework given below, one has to know the outline of these theoretical backgrounds.

2.1 Definition of the cross section

The original *unrenormalized* Lagrangian density is divided into free and interaction parts as

$$\mathcal{L}(x) = \mathcal{L}_{free0}(x) + \mathcal{L}_{int0}(x), \tag{2.1}$$

where free part contains all the quadratic form of fields including gauge fixing term. Since all the models we are considering are renormalizable, the Lagrangian can be reexpressed in terms of the *renormalized* quantities. Thus we can write

$$\mathcal{L}(x) = \mathcal{L}_{free}(x) + \mathcal{L}_{int}(x) + \delta\mathcal{L}_c(x). \tag{2.2}$$

Here the last term represents the counterterm Lagrangian(in the current version of GRACE this part is of no use, because no loop amplitude can be generated).

Denoting the substantial interaction as

$$\tilde{\mathcal{L}}_{int}(x) \equiv \mathcal{L}_{int}(x) + \delta\mathcal{L}_c(x) \tag{2.3}$$

we define the S-matrix,

$$S = T^* \exp[i \int \mathrm{d}^4x \tilde{\mathcal{L}}_{int}(x)], \tag{2.4}$$

where T^* is the usual chronological operator, introduced when $\tilde{\mathcal{L}}_{int}$ contains derivative of fields. Expanding the exponential, we have a perturbative series with respect to the interaction Lagrangian

$$S = 1 + \sum_{N=1}^{\infty} \frac{i^N}{N!} \int \mathrm{d}^4 x_1 \cdots \int \mathrm{d}^4 x_N T^* [\tilde{\mathcal{L}}_{int}(x_1) \tilde{\mathcal{L}}_{int}(x_2) \cdots \tilde{\mathcal{L}}_{int}(x_N)]. \tag{2.5}$$

The scattering matrix T is defined by an operator relation

$$S = 1 + iT. \tag{2.6}$$

By taking the matrix element between initial and final states $|i\rangle$ and $|f\rangle$ we have

$$S_{fi} = \delta_{fi} + i(2\pi)^4 \delta^4(P_f - P_i) T_{fi}, \tag{2.7}$$

where P_i and P_f are the total four momenta of initial and final states, respectively. The cross section is defined as

$$\sigma = \frac{1}{\mathrm{flux}} \int \mathrm{d}\Gamma_f (2\pi)^4 \delta^4(P_f - P_i) \sum_{\mathrm{spin}} |T_{fi}|^2. \tag{2.8}$$

Here 'flux' is the flux of incident particles and $\mathrm{d}\Gamma_f$ is the volume element of phase space of the final states. In our convention we define this element for any kind of particle as

$$\mathrm{d}\Gamma_f = \prod_{i=1}^{N_f} \frac{\mathrm{d}^3 \boldsymbol{q}_i}{2q_{0i}(2\pi)^3}, \tag{2.9}$$

where $q_i = (q_{i0}, \boldsymbol{q}_i), \quad i = 1, \cdots, N_f$ are the four-momenta of N_f particles in the final state. Then the initial flux is normalized as

$$\mathrm{flux} = v_{\mathrm{rel}} 2p_{10} 2p_{20}, \tag{2.10}$$

where p_{10} and p_{20} are energies of incoming two particles and v_{rel} is the relative velocity of these two.
Thus the final form of the cross section for the process

$$p_1 + p_2 \to q_1 + q_2 + \cdots + q_{N_f}, \tag{2.11}$$

is given by

$$\sigma = \frac{1}{v_{\mathrm{rel}} 2p_{10} 2p_{20}} \int \prod_{i=1}^{N_f} \frac{\mathrm{d}^3 \boldsymbol{q}_i}{2q_{i0}(2\pi)^3} (2\pi)^4 \delta^4 \left(p_1 + p_2 - \sum_{i=1}^{N_f} q_i \right) \sum_{h_f} \overline{\sum_{h_i}} |T_{fi}|^2. \tag{2.12}$$

Here helicity states of final particles(h_f) are summed and those of initial state(h_i) are averaged for the simplest case.
Though the FORTRAN output from `GRACE` automatically provides the quantity

$$\tau_{fi} = \sum_{h_f} \overline{\sum_{h_i}} |T_{fi}|^2, \tag{2.13}$$

as the default output, one can select any helicity state in both initial and final states by changing the part of the program corresponding to these summations as explained in section 3.2.1.
If the incident particles collide in the center-of-mass system, the flux factor is given by

$$\text{flux} \simeq 2s, \tag{2.14}$$

neglecting their masses, where $s = (p_1 + p_2)^2$ is the square of total energy. When these particles are partons with energy fractions x_1 and x_2, like in pp or $p\bar{p}$ colliders, it is given by

$$\text{flux} \simeq 2(x_1 x_2)s, \tag{2.15}$$

and the cross section Eq.(2.12) is that of sub-process.

2.2 Metric and conventions

I) **Convention for Feynman rules.**

The scattering amplitude T_{fi} is constructed according to the Feynman rules. There are some different ways to decompose the factor i^N in Eq.(2.5) and assign to various parts of a diagram. The convention we use in `GRACE` is the following:

1) Let us denote a generic field as ϕ. The propagator is defined by

$$D_F(p) = i \int \mathrm{d}^4x \ \mathrm{e}^{-ip\cdot x} \langle 0|T(\phi(x)\phi^\dagger(0))|0\rangle. \tag{2.16}$$

Thus for fermion we have

$$S_F(p) = \frac{1}{-\not{p} + m - i\varepsilon}, \tag{2.17}$$

for scalar particle

$$\Delta_F(p) = \frac{1}{-p^2 + m^2 - i\varepsilon}, \tag{2.18}$$

and for gauge boson of mass M

$$D_{F\mu\nu}(q) = \frac{G_{\mu\nu}(q)}{-q^2 + M^2 - i\varepsilon}, \tag{2.19}$$

where $G_{\mu\nu}(q)$ is a symmetric tensor which depends on the gauge condition used. Its explicit form will appear in section 2.3.

2) The vertex is defined as the Fourier transform of the interaction Lagrangian,

$$\int \mathrm{d}^4x \ \mathrm{e}^{-ip\cdot x} \tilde{\mathcal{L}}_{int}(x). \tag{2.20}$$

For example, photon vertex of charged fermion f is given by

$$eQ_f\gamma_\mu. \tag{2.21}$$

3) For a loop diagram with L loops we assign the following loop integration(in n-dimension)

$$\int \prod_{l=1}^{L} \frac{\mathrm{d}^n k_l}{(2\pi)^n i}. \tag{2.22}$$

By this convention we can save the number of i because after making Wick's rotation each loop integration will produce an i to compensate that in the denominator. Simple counting will show that i^N in Eq.(2.5) can be divided into three parts

$$i^N = i \cdot i^{N+L-1} \cdot i^{-L}, \tag{2.23}$$

where the first i is regarded as that of iT, the second is absorbed into the definition of propagators and the last into loop integrals.

II) **Metric**

The metric convention is as follows;

1) the space-time metric $g_{\mu\nu}$ is defined by

$$g_{00} = 1, \quad g_{ij} = -\delta_{ij} \qquad \text{for} \quad i,j = 1,2,3, \tag{2.24}$$

2) the components of a four-momentum is given by

$$p = (p_0, \boldsymbol{p}), \tag{2.25}$$

and the inner product of two arbitrary four-momenta, p and q, is

$$p \cdot q = p_0 q_0 - \boldsymbol{p} \cdot \boldsymbol{q}. \tag{2.26}$$

3) the 4-dimensional Dirac matrix satisfies

$$\gamma_\mu \gamma_\nu + \gamma_\nu \gamma_\mu = 2 g_{\mu\nu}, \qquad \mu, \nu = 0, \cdots, 3 \tag{2.27}$$

and γ_5 is defined by

$$\gamma_5 = i \gamma_0 \gamma_1 \gamma_2 \gamma_3, \tag{2.28}$$

as usual. The hermite conjugate of γ-matrix obeys

$$\gamma_0 \gamma_\mu^\dagger \gamma_0 = \gamma_\mu, \tag{2.29}$$

hence

$$\gamma_0 \gamma_5^\dagger \gamma_0 = -\gamma_5. \tag{2.30}$$

In `GRACE` system, `CHANEL` calculates Dirac matrices in a numerical way, but specification of their explicit representations is *not* necessary.

III) Normalization of wave function.

1) Massive Dirac spinor
The Dirac spinor is normalized as

$$\begin{aligned}\overline{u}(p,s)\gamma_0 u(p,s') &= 2p_0\delta_{ss'},\\ \overline{v}(p,s)\gamma_0 v(p,s') &= 2p_0\delta_{ss'},\end{aligned} \tag{2.31}$$

hence the projection operators are

$$\begin{aligned}u(p,s)\overline{u}(p,s) &= \frac{1+\gamma_5 \not{s}}{2}(\not{p}+m),\\ v(p,s)\overline{v}(p,s) &= \frac{1-\gamma_5 \not{s}}{2}(\not{p}-m),\end{aligned} \tag{2.32}$$

where $u(p,s)$ and $v(p,s)$ are spinors of particle and anti-particle with momentum p and spin vector s. The latter satisfies

$$s^2 = -1, \quad s\cdot p = 0. \tag{2.33}$$

When one specifies a helicity state, the spin vector has the form

$$s = h\cdot\left(\frac{|\boldsymbol{p}|}{m}, \frac{p_0}{m}\boldsymbol{n}\right), \quad \boldsymbol{n} = \frac{\boldsymbol{p}}{|\boldsymbol{p}|} \tag{2.34}$$

with $h = \pm 1$.

2) Massless Dirac spinor
As massless fermion, we know only left-handed neutrinos. Denoting its spinors as $u_\nu(p)$ and $v_\nu(p)$, the projection operators are then

$$\begin{aligned}u_\nu(p)\overline{u}_\nu(p) &= \frac{1-\gamma_5}{2}\not{p},\\ v_\nu(p)\overline{v}_\nu(p) &= \frac{1+\gamma_5}{2}\not{p}.\end{aligned} \tag{2.35}$$

3) Spin summation of massless gauge boson.
Polarization vector of photon or gluon $\epsilon^{(\lambda)}_\mu(k)$, $\lambda = 1,2$, $k^2 = 0$, satisfies

$$k\cdot\epsilon^{(\lambda)}(k) = 0, \quad \epsilon^{(\lambda)}(k)\cdot\epsilon^{(\lambda')}(k) = -\delta_{\lambda\lambda'}, \tag{2.36}$$

spin summation is given by

$$\sum_{\lambda=1}^{2}\epsilon^{(\lambda)}_\mu(k)\epsilon^{(\lambda)}_\nu(k) = -g_{\mu\nu} + \frac{k_\mu n_\nu + k_\nu n_\mu}{k\cdot n} - n^2\frac{k_\mu k_\nu}{(k\cdot n)^2}, \tag{2.37}$$

where n is an arbitrary constant vector. As `CHANEL` uses helicity formalism, it defines an expression of $\epsilon^{(\lambda)}_\mu(k)$ and the spin summation is consistent with this formula as shown in section 2.4.

4) Spin summation of massive gauge boson.
Denoting the polarization vector of $W^{\pm}$ or Z^0 as $\epsilon_\mu^{(\lambda)}(k)$, $\lambda = 1, 2, 3$, $k^2 = M^2$, we have

$$k \cdot \epsilon^{(\lambda)}(k) = 0, \quad \epsilon^{(\lambda)}(k) \cdot \epsilon^{(\lambda')}(k) = -\delta_{\lambda\lambda'}, \tag{2.38}$$

and the spin summation

$$\sum_{\lambda=1}^{3} \epsilon_\mu^{(\lambda)}(k) \epsilon_\nu^{(\lambda)}(k) = -g_{\mu\nu} + \frac{k_\mu k_\nu}{M^2}. \tag{2.39}$$

2.3 Specification of models

Now we turn to the details of the models prepared in the system. In the present version of `GRACE` we have included only standard ones;

- QED
- Electroweak
- QCD

Although the first one is, of course, a part of the second, the system is designed so that one can choose pure QED. The models are selected by giving the order of coupling of each model. It should be noted that the orders of couplings for electroweak and QED models are used exclusively, *i.e.* they should not be given at the same time. (If one generates amplitudes in electroweak theory, then one can choose only QED part by using diagram selection flag.) Since the current version of `GRACE` can provide only tree amplitudes, the counterterm is not needed at present stage but we described the whole renormalized Lagrangian in anticipating the forthcoming version which includes 1-loop diagrams.

2.3.1 QED

The unrenormalized Lagrangian density can be divided into

$$\mathcal{L}_{QED} = \mathcal{L}_{free0} + \mathcal{L}_{int0}. \tag{2.40}$$

The free Lagrangian $\mathcal{L}_{free0}$ is

$$\mathcal{L}_{free0} = \sum_f \overline{\psi}_0^{(f)} (i\gamma \cdot \partial - m_{f0}) \psi_0^{(f)} - \frac{1}{4} F_{\mu\nu 0} F_0^{\mu\nu} + \mathcal{L}_{gauge}, \tag{2.41}$$

where f indicates the fermion f and 0 means unrenormalized quantity. Note that the summation over f also implies the sum over color degree of freedom for quarks. The interaction part is given by

$$\mathcal{L}_{\text{int}0} = \sum_f e_0 Q_f \overline{\psi}_0^{(f)} \gamma_\mu \psi_0^{(f)} A_0^\mu. \tag{2.42}$$

The electromagnetic charge of a fermion f is given by $e_0 Q_f$ with the positron charge e_0. Thus up-quark has $Q_{up} = +2/3$, down-quark $Q_{down} = -1/3$, electron $Q_e = -1$, and so on. The gauge fixing term, written in renormalized fields, is

$$\begin{aligned} \mathcal{L}_{gauge} &= -\frac{1}{2\alpha}(\partial \cdot A)^2, && \text{covariant gauge} \\ \mathcal{L}_{gauge} &= -\frac{1}{2\alpha}(n \cdot A)^2, && \text{axial gauge} \end{aligned} \tag{2.43}$$

where α is a gauge parameter and n is an arbitrary constant vector. If $n^2 = 0$, then it is called *light-cone* gauge.

Introducing renormalization constants Z_{1f}, Z_{2f}, Z_3 and δm_f and replacing all the quantities in this Lagrangian by renormalized ones,

$$\begin{aligned} \psi_{f0} &= Z_{2f}\psi_f, \quad A_{\mu 0} = Z_3 A_\mu, \\ e_0 &= e Z_{1f} Z_{2f}^{-1} Z_3^{-1/2}, \\ m_{f0} &= m_f + \delta m_f, \end{aligned} \tag{2.44}$$

we can rewrite the original Lagrangian by renormalized quantities and we have

$$\mathcal{L}_{QED} = \mathcal{L}_{free} + \mathcal{L}_{int} + \delta\mathcal{L}_c, \tag{2.45}$$

with

$$\begin{aligned} \mathcal{L}_{free} &= \sum_f \overline{\psi}^{(f)} \left(i\partial\!\!\!/ - m_f\right) \psi^{(f)} - \frac{1}{4} F_{\mu\nu}F^{\mu\nu} + \mathcal{L}_{gauge} \\ \mathcal{L}_{int} &= \sum_f e Q_f \overline{\psi}^{(f)} \gamma_\mu \psi^{(f)} A^\mu \\ \delta\mathcal{L}_c &= \sum_f \delta Z_{2f} \overline{\psi}^{(f)} \left(i\partial\!\!\!/ - m_f\right) \psi^{(f)} - \sum_f Z_{2f}\delta m_f \overline{\psi}^{(f)} \psi^{(f)} \\ &\quad - \frac{1}{4}\delta Z_3 F_{\mu\nu}F^{\mu\nu} + \sum_f e Q_f \delta Z_{1f} \overline{\psi}^{(f)} \gamma_\mu \psi^{(f)} A^\mu. \end{aligned} \tag{2.46}$$

Here the counterterm Lagrangian contains

$$\begin{aligned} Z_{2f} &= 1 + \delta Z_{2f} \\ Z_3 &= 1 + \delta Z_3 \\ Z_{1f} &= 1 + \delta Z_{1f}. \end{aligned} \tag{2.47}$$

Note that the gauge invariance or charge universality implies

$$Z_{1f} \equiv Z_{2f}. \tag{2.48}$$

The renormalization conditions to fix these constant are well known, and will be found any standard textbook of quantum field theory.
The propagators are as follows; for fermion it is given by Eq.(2.17), for photon,

$$D_{\mu\nu}(q) = \frac{G_{\mu\nu}(q)}{-q^2 - i\varepsilon}. \tag{2.49}$$

The numerator takes the following forms depending on the gauge condition:

$$G_{\mu\nu}(q) = -g_{\mu\nu} + (1-\alpha)\frac{q_\mu q_\nu}{q^2}, \qquad \text{covariant gauge} \tag{2.50}$$

$$G_{\mu\nu}(q) = -g_{\mu\nu} + \frac{q_\mu n_\nu + q_\nu n_\mu}{q \cdot n} - (n^2 + \alpha g^2)\frac{q_\mu q_\nu}{(q\cdot n)^2}. \qquad \text{axial gauge} \tag{2.51}$$

2.3.2 Electroweak theory

The standard model of $SU(2)_L \times U(1)$ gauge theory, originally proposed by Glashaw-Weinberg-Salam, is much more complicated than QED. Quarks and leptons are classified into left- and right-handed, which transform under the gauge group in a different way;

$$\begin{pmatrix}\nu_e\\ e\end{pmatrix}_L, \quad \begin{pmatrix}\nu_\mu\\ \mu\end{pmatrix}_L, \quad \begin{pmatrix}\nu_\tau\\ \tau\end{pmatrix}_L, \quad e_R, \quad \mu_R, \quad \tau_R,$$

$$\begin{pmatrix}u\\ d\end{pmatrix}_L, \quad \begin{pmatrix}c\\ s\end{pmatrix}_L, \quad \begin{pmatrix}t\\ b\end{pmatrix}_L, \quad u_R, \quad d_R, \quad c_R, \quad s_R, \quad t_R, \quad b_R.$$

Two kinds of gauge boson fields are introduced which transform as $SU(2)$-triplet and -singlet,

$$\begin{aligned} \text{triplet} &\longrightarrow A^1_\mu(x), A^2_\mu(x), A^3_\mu(x) \\ \text{singlet} &\longrightarrow B_\mu(x). \end{aligned}$$

The Higgs field is also a doublet

$$\Phi(x) = \frac{1}{\sqrt{2}}\begin{pmatrix} i\phi^+(x) \\ \phi^0(x)\end{pmatrix}. \tag{2.52}$$

Before giving the explicit form of the Lagrangian, we like to make a comment on the parameters of the theory.

Constant parameters

The most fundamental constants in the Lagrangian are two coupling constants of $SU(2)$ and $U(1)$ gauge interactions(g and g', respectively) and the vacuum expectation value of the neutral Higgs scalar,

$$g, g', \langle\phi^0\rangle.$$

In the classical level the heavy boson masses and the electric charge are given by

$$\begin{aligned} M_Z^2 &= \frac{1}{4}(g^2+g'^2)\langle\phi^0\rangle^2, \quad M_W^2 = \frac{1}{4}g^2\langle\phi^0\rangle^2, \\ e &= \frac{gg'}{\sqrt{g^2+g'^2}}, \end{aligned} \tag{2.53}$$

respectively. Thus the alternative set of parameters is

$$e, M_W, M_Z,$$

which are physically observable quantities. The weak mixing angle $\sin^2\theta_W$ is defined through the relation

$$\sin^2\theta_W = 1 - \frac{M_W^2}{M_Z^2}. \tag{2.54}$$

In our convention e, M_W and M_Z are used as input parameters. However, as the precise value of $W^\pm$ boson mass has not yet been measured, the muon decay width, Γ_μ, is more reliable than M_W at present. Using the above set of constants one can express the width in a form (up to any order of perturbation)

$$\Gamma_\mu = M_W \cdot f(\alpha, M_W^2/M_Z^2), \tag{2.55}$$

(with possible dependence on Higgs and t-quark masses, m_H and m_t). Solving this equation and using the experimental value for Γ_μ, we can get M_W as a function of other parameters,

$$M_W = M_Z \cdot h(\alpha, \Gamma_\mu/M_Z). \tag{2.56}$$

In this sense the set of constants

$$e, \Gamma_\mu, M_Z,$$

can be used as the input parameters of the theory.

Lagrangian

We follow the formulation given in Ref.[1]. As the full Lagrangian has very complicated structure, we divide it into two parts; the first has the same form as the classical Lagrangian containing physical objects and the second is related to the gauge fixing,

$$\mathcal{L}_{ELW} = \mathcal{L}_{cl} + \mathcal{L}_{gauge}. \tag{2.57}$$

The first part is further decomposed into several terms,

$$\mathcal{L}_{cl} = \mathcal{L}_{G0} + \mathcal{L}_{F0} + \mathcal{L}_{H0} + \mathcal{L}_{M0}, \tag{2.58}$$

where $\mathcal{L}_{G0}$ is the gauge boson part, $\mathcal{L}_{F0}$ the fermion kinetic part, $\mathcal{L}_{H0}$ the Higgs scalar part and $\mathcal{L}_{M0}$ the fermion-Higgs interaction.

1) The pure gauge boson part contains only $SU(2)$ and $U(1)$ gauge fields,

$$\mathcal{L}_{G0} = -\frac{1}{4}G^a_{\mu\nu 0}G^a_{\mu\nu 0} - \frac{1}{4}F_{\mu\nu 0}F_{\mu\nu 0}, \tag{2.59}$$

where

$$\begin{aligned} G^a_{\mu\nu 0} &= \partial_\mu A^a_{\nu 0} - \partial_\nu A^a_{\mu 0} + g_0\epsilon_{abc}A^b_{\mu 0}A^c_{\nu 0}, \\ F_{\mu\nu 0} &= \partial_\mu B_{\nu 0} - \partial_\nu B_{\mu 0}, \end{aligned}$$

are field strengths for gauge fields $A^a_{\mu 0}(a = 1,2,3)$ and $B_{\mu 0}$, respectively.

2) The kinetic part of fermions, both quark and leptons, including gauge interactions is given by,

$$\mathcal{L}_{F0} = \sum_L \bar{\psi}_{L0}(i\not{\partial} + g_0 T_a A^a_{\mu 0} + g'_0 T_0 B_{\mu 0})\psi_{L0} + \sum_{f=i,I} \bar{\psi}^{(f)}_{R0}\,(i\not{\partial} + g'_0 T_0 B_{\mu 0})\,\psi^{(f)}_{R0}, \tag{2.60}$$

where ψ_{L0} and ψ_{R0} represent $SU(2)$ doublet and singlet fermion fields, respectively, with

$$\psi_{L0} = \begin{pmatrix} \psi^{(I)}_{L0} \\ \psi^{(i)}_{L0} \end{pmatrix}. \tag{2.61}$$

To specify a fermion we use the subscript (L) and the superscripts (I), (i) and (f) which stand for left-handed fermion doublet and upper, lower and all kinds of fermion, respectively. The coupling constant g_0 corresponds to $SU(2)$ and g'_0 to $U(1)$ gauge interactions and T_a's are related to $SU(2)$ Pauli matrices, $T_a = \tau_a/2.(a = 1,2,3)$ and $T_0 = Q - T_3$ where Q is the charge operator.

3) The Higgs scalar part with gauge interaction is

$$\mathcal{L}_{H0} = \left|\left(\partial_\mu - ig_0 T_a A^a_{\mu 0} - i\frac{g'_0}{2}B_{\mu 0}\right)\Phi_0\right|^2 + \mu_0^2\Phi_0^\dagger\Phi_0 - \lambda_0(\Phi_0^\dagger\Phi_0)^2. \tag{2.62}$$

4) The fermion-Higgs interaction is

$$\mathcal{L}_{M0} = -\sum_i f^{(i)}_0 \bar{\Psi}^{(i)}_{L0}\Phi_0\psi^{(i)}_{R0} - \sum_I f^{(I)}_0 \bar{\Psi}^{(I)}_{L0}(i\tau_2\Phi_0^*)\psi^{(I)}_{R0} + h.c., \tag{2.63}$$

where $f^{(i)}_0$ and $f^{(I)}_0$ are Yukawa coupling constants. Two new combinations of fields, $\Psi^{(i)}_{L0}$ and $\Psi^{(I)}_{L0}$, for left-handed fermions are introduced so as to make the mass matrix diagonal,

$$\begin{aligned} \Psi^{(i)}_{L0} &= \begin{pmatrix} \sum_I U_{iI}\psi^{(I)}_{L0} \\ \psi^{(i)}_{L0} \end{pmatrix}, \\ \Psi^{(I)}_{L0} &= \begin{pmatrix} \psi^{(I)}_{L0} \\ \sum_i U^{-1}_{Ii}\psi^{(i)}_{L0} \end{pmatrix}, \end{aligned} \tag{2.64}$$

where U_{iI} is the mixing matrix for quarks. In the current version of `GRACE` this mixing is *not supported*, namely the unit matrix is assumed for U_{iI}.

The symmetries are broken by the vacuum expectation value of the Higgs scalar field Φ_0,

$$\Phi_0 = \frac{1}{\sqrt{2}} \begin{pmatrix} i\chi_0^+ \\ v_0 + \phi_0 - i\chi_{30} \end{pmatrix}. \tag{2.65}$$

Here,

1) v_0 is the bare vacuum expectation value,

2) ϕ_0 is the physical Higgs scalar field,

3) χ_{30} is the neutral Goldstone boson,

4) χ_0^+ is the charged Goldstone boson, defined as

$$\chi_0^\pm = \frac{1}{\sqrt{2}}(\chi_{10} \mp i\chi_{20}). \tag{2.66}$$

We introduce the physical gauge boson fields by

$$\begin{aligned} W_{\mu 0}^\pm &= \frac{1}{\sqrt{2}}(A_{\mu 0}^1 \mp i A_{\mu 0}^2), \\ Z_{\mu 0} &= \frac{1}{\sqrt{g_0^2 + g_0'^2}}(g_0 A_{\mu 0}^3 - g_0' B_{\mu 0}), \\ A_{\mu 0} &= \frac{1}{\sqrt{g_0^2 + g_0'^2}}(g_0' A_{\mu 0}^3 + g_0 B_{\mu 0}). \end{aligned} \tag{2.67}$$

Collecting all of these, we get unrenormalized Lagrangian expressed in terms of physical particles. The bosonic part becomes

$$\begin{aligned} \mathcal{L}_{G0} = & -\frac{1}{2}|\partial_\mu W_{\nu 0}^+ - \partial_\nu W_{\mu 0}^+|^2 - \frac{1}{4}(\partial_\mu Z_{\nu 0} - \partial_\nu Z_{\mu 0})^2 - \frac{1}{4}(\partial_\mu A_{\nu 0} - \partial_\nu A_{\mu 0})^2 \\ & + \frac{i g_0}{\sqrt{g_0^2 + g_0'^2}}(g_{\alpha\gamma} g_{\beta\delta} - g_{\alpha\delta} g_{\beta\gamma})[g_0\{(\partial_\alpha W_{\beta 0}^+) W_{\gamma 0}^- Z_{\delta 0} + (\partial_\alpha W_{\beta 0}^-) Z_{\gamma 0} W_{\delta 0}^+ \\ & + (\partial_\alpha Z_{\beta 0}) W_{\gamma 0}^+ W_{\delta 0}^-\} \\ & + g_0'\{(\partial_\alpha W_{\beta 0}^+) W_{\gamma 0}^- A_{\delta 0} + (\partial_\alpha W_{\beta 0}^-) A_{\gamma 0} W_{\delta 0}^+ + (\partial_\alpha A_{\beta 0}) W_{\gamma 0}^+ W_{\delta 0}^-\}] \\ & - \frac{g_0^2}{g_0^2 + g_0'^2}[(g_{\alpha\beta} g_{\gamma\delta} - g_{\alpha\gamma} g_{\beta\delta}) W_{\alpha 0}^+ W_{\beta 0}^- (g_0^2 Z_{\gamma 0} Z_{\delta 0} + g_0'^2 A_{\gamma 0} A_{\delta 0})] \\ & + (2 g_{\alpha\beta} g_{\gamma\delta} - g_{\alpha\gamma} g_{\beta\delta} - g_{\alpha\delta} g_{\beta\gamma}) g_0 g_0' W_{\alpha 0}^+ W_{\beta 0}^- A_{\gamma 0} Z_{\delta 0} \\ & + (g_{\alpha\beta} g_{\gamma\delta} - g_{\alpha\gamma} g_{\beta\delta}) \frac{g_0^2}{2} W_{\alpha 0}^+ W_{\beta 0}^+ W_{\gamma 0}^- W_{\delta 0}^-, \end{aligned} \tag{2.68}$$

and the fermionic part is

$$\begin{aligned}\mathcal{L}_{F0} &= \sum_f i\bar{\psi}_0^{(f)}\partial\!\!\!/\psi_0^{(f)} + e_0\sum_f Q_f\bar{\psi}_0^{(f)}\gamma_\mu\psi_0^{(f)}A_{\mu 0} \\ &+\frac{g_0}{\sqrt{2}}\sum_{i,I}\left(\bar{\psi}_0^{(I)}U_{Ii}^\dagger\gamma_\mu\frac{1-\gamma_5}{2}\psi_0^{(i)}W_{\mu 0}^+ + \bar{\psi}_0^{(i)}U_{iI}\gamma_\mu\frac{1-\gamma_5}{2}\psi_0^{(I)}W_{\mu 0}^-\right) \\ &+\frac{\sqrt{g_0^2+g_0'^2}}{2}\sum_f\bar{\psi}_0^{(f)}\gamma_\mu\Big[T_{3f}(1-\gamma_5)-2Q_f\frac{g_0'^2}{g_0^2+g_0'^2}\Big]\psi_0^{(f)}Z_{\mu 0},\end{aligned} \tag{2.69}$$

where ψ is the Dirac spinor which we define

$$\psi_0 = \psi_{L0}+\psi_{R0} = \frac{1-\gamma_5}{2}\psi_0 + \frac{1+\gamma_5}{2}\psi_0. \tag{2.70}$$

Further we have introduced the bare electric charge by

$$e_0 = \frac{g_0 g_0'}{\sqrt{g_0^2+g_0'^2}}, \tag{2.71}$$

and Q_f is the charge of f-th fermion in the unit of positron charge e_0. As the Higgs scalar part becomes complicated, we divide it further into four parts,

$$\mathcal{L}_{H0} = \mathcal{L}_{H0}^{(2)} + \mathcal{L}_{H0}^{(3)} + \mathcal{L}_{H0}^{(4)} + \mathcal{L}_{V0}. \tag{2.72}$$

The first part contains kinetic terms of scalar particles and bilinear terms in fields,

$$\begin{aligned}\mathcal{L}_{H0}^{(2)} &= +\frac{1}{2}(\partial_\mu\phi_0)^2 + \frac{1}{2}(\partial_\mu\chi_{30})^2 + (\partial_\mu\chi_0^+)(\partial_\mu\chi_0^-) \\ &+M_{W0}^2W_{\mu 0}^+W_{\mu 0}^- + \frac{1}{2}M_{Z0}^2Z_{\mu 0}Z_{\mu 0} \\ &-\frac{1}{2}g_0v_0[W_{\mu 0}^+(\partial_\mu\chi_0^-)+W_{\mu 0}^-(\partial_\mu\chi_0^+)] - \frac{1}{2}\sqrt{g_0^2+g_0'^2}v_0Z_{\mu 0}(\partial_\mu\chi_{30}),\end{aligned} \tag{2.73}$$

where the bare mass terms of gauge bosons are introduced by

$$\begin{aligned}M_{Z0}^2 &= \frac{1}{4}(g_0^2+g_0'^2)v_0^2, \\ M_{W0}^2 &= \frac{1}{4}g_0^2v_0^2.\end{aligned} \tag{2.74}$$

The cubic and quartic parts in fields contain interaction terms between Higgs and gauge fields,

$$\begin{aligned}\mathcal{L}_{H0}^{(3)} &= \frac{1}{2}g_0W_{\mu 0}^+(\chi_0^- \overleftrightarrow{\partial_\mu} \phi_0) + \frac{1}{2}g_0W_{\mu 0}^-(\chi_0^+ \overleftrightarrow{\partial_\mu} \phi_0) \\ &+\frac{i}{2}g_0W_{\mu 0}^+(\chi_{30} \overleftrightarrow{\partial_\mu} \chi_0^-) - \frac{i}{2}g_0W_{\mu 0}^-(\chi_{30} \overleftrightarrow{\partial_\mu} \chi_0^+) + \frac{1}{2}\sqrt{g_0^2+g_0'^2}Z_{\mu 0}(\chi_{30} \overleftrightarrow{\partial_\mu} \phi_0) \\ &+\frac{i}{2}\frac{g_0^2-g_0'^2}{\sqrt{g_0^2+g_0'^2}}Z_{\mu 0}(\chi_0^- \overleftrightarrow{\partial_\mu} \chi_0^+) + \frac{ig_0g_0'}{\sqrt{g_0^2+g_0'^2}}A_{\mu 0}(\chi_0^- \overleftrightarrow{\partial_\mu} \chi_0^+),\end{aligned} \tag{2.75}$$

$$\begin{aligned}
\mathcal{L}_{H0}^{(4)} &= \frac{1}{4}g_0^2 W_{\mu 0}^+ W_{\mu 0}^- (2v_0\phi_0 + \phi_0^2 + 2\chi_0^+\chi_0^- + \chi_{30}^2) \\
&+\frac{1}{4}\frac{(g_0^2 - g_0'^2)^2}{\sqrt{g_0^2+g_0'^2}} Z_{\mu 0}Z_{\mu 0}(\chi_0^+\chi_0^-) + \frac{1}{8}(g_0^2+g_0'^2)Z_{\mu 0}Z_{\mu 0}(2v_0\phi_0 + \phi_0^2 + \chi_{30}^2) \\
&+\frac{g_0^2 g_0'^2}{g_0^2+g_0'^2}A_{\mu 0}A_{\mu 0}(\chi_0^+\chi_0^-) + \frac{g_0 g_0'(g_0^2 - g_0'^2)}{g_0^2+g_0'^2}A_{\mu 0}Z_{\mu 0}(\chi_0^+\chi_0^-) \\
&+\frac{1}{2}\frac{g_0 g_0'^2}{\sqrt{g_0^2+g_0'^2}}(Z_{\mu 0}W_{\mu 0}^+\chi_0^-)(\chi_{30} + iv_0 + i\phi_0) \\
&+\frac{1}{2}\frac{g_0 g_0'^2}{\sqrt{g_0^2+g_0'^2}}(Z_{\mu 0}W_{\mu 0}^-\chi_0^+)(\chi_{30} - iv_0 - i\phi_0) \\
&-\frac{1}{2}\frac{g_0^2 g_0'}{\sqrt{g_0^2+g_0'^2}}(A_{\mu 0}W_{\mu 0}^+\chi_0^-)(\chi_{30} + iv_0 + i\phi_0) \\
&-\frac{1}{2}\frac{g_0^2 g_0'}{\sqrt{g_0^2+g_0'^2}}(A_{\mu 0}W_{\mu 0}^-\chi_0^+)(\chi_{30} - iv_0 - i\phi_0). \qquad (2.76)
\end{aligned}$$

The last part is the potential term for Higgs particles,

$$\begin{aligned}
\mathcal{L}_{V0} &= v_0(\mu_0^2 - \lambda_0 v_0^2)\phi_0 + (\mu_0^2 - \lambda_0 v_0^2)\chi_0^+\chi_0^- \\
&+\frac{1}{2}(\mu_0^2 - \lambda_0 v_0^2)\chi_{30}^2 + \frac{1}{2}(\mu_0^2 - 3\lambda_0 v_0^2)\phi_0^2 \\
&-2v_0\lambda_0(\phi_0\chi_0^+\chi_0^-) - v_0\lambda_0(\phi_0\chi_{30}\chi_{30}) - v_0\lambda_0\phi_0^3 \\
&-\lambda_0(\chi_0^+\chi_0^-)^2 - \lambda_0(\chi_0^+\chi_0^-)\chi_{30}^2 - \lambda_0(\chi_0^+\chi_0^-)\phi_0^2 \\
&-\frac{1}{4}\lambda_0\chi_{30}^4 - \frac{1}{4}\lambda_0\phi_0^4 - \frac{1}{2}\lambda_0\chi_{30}^2\phi_0^2. \qquad (2.77)
\end{aligned}$$

The fermion mass part yields,

$$\begin{aligned}
\mathcal{L}_{M0} &= -\sum_f m_{f0}\overline{\psi}_0^{(f)}\psi_0^{(f)} \\
&-\frac{i}{2}\sum_{i,I}\overline{\psi}_0^{(I)}U_{Ii}^\dagger[(f_0^{(i)} - f_0^{(I)}) + (f_0^{(i)} + f_0^{(I)})\gamma_5]\psi_0^{(i)}\chi_0^+ \\
&-\frac{i}{2}\sum_{i,I}\overline{\psi}_0^{(i)}U_{iI}[(f_0^{(I)} - f_0^{(i)}) + (f_0^{(i)} + f_0^{(I)})\gamma_5]\psi_0^{(I)}\chi_0^- \\
&-\sum_f \frac{f_0^{(f)}}{\sqrt{2}}\phi_0\overline{\psi}_0^{(f)}\psi_0^{(f)}
\end{aligned}$$

$$+\sum_i \frac{if_0^{(i)}}{\sqrt{2}}\chi_{30}\overline{\psi}_0^{(i)}\gamma_5\psi_0^{(i)} - \sum_I \frac{if_0^{(I)}}{\sqrt{2}}\chi_{30}\overline{\psi}_0^{(I)}\gamma_5\psi_0^{(I)}. \tag{2.78}$$

Next we turn to the gauge fixing of the original Lagrangian. This contains unphysical particles, Goldstone bosons and ghosts, that is, the gauge fixing term and the Faddeev-Poppov ghost parts,

$$\mathcal{L}_{gauge} = \mathcal{L}_{GF} + \mathcal{L}_{FP}. \tag{2.79}$$

The gauge fixing term is written in the renormalized form as

$$\begin{aligned}\mathcal{L}_{GF} &= -\frac{1}{\alpha_W}(\partial_\mu W_\mu^+ + \alpha_W M_W \chi^+)\cdot(\partial_\mu W_\mu^- + \alpha_W M_W \chi^-) \\ &\quad -\frac{1}{2\alpha_Z}(\partial_\mu Z_\mu + \alpha_Z M_Z \chi_3)^2 - \frac{1}{2\alpha_A}(\partial_\mu A_\mu)^2.\end{aligned} \tag{2.80}$$

The last part, Faddeev-Poppov ghost, is given by

$$\begin{aligned}\mathcal{L}_{FP} &= -\bar{c}_0^+\delta_{BRS}(\partial_\mu W_{\mu0}^- + \alpha_{W0}M_{W0}\chi_0^-) - \bar{c}_0^-\delta_{BRS}(\partial_\mu W_{\mu0}^+ + \alpha_{W0}M_{W0}\chi_0^+) \\ &\quad -\bar{c}_0^Z\delta_{BRS}(\partial_\mu Z_{\mu0} + \alpha_{Z0}M_{Z0}\chi_{30}) - \bar{c}_0^A\delta_{BRS}(\partial_\mu A_{\mu0} + \beta_0 M_{Z0}\chi_{30}),\end{aligned} \tag{2.81}$$

where every quantity is bare and the BRS transformations for the fields are defined by

$$\begin{aligned}\delta_{BRS}W_{\mu0}^\pm &= \partial_\mu c_0^\pm \pm \frac{ig_0}{\sqrt{g_0^2+g_0'^2}}[W_{\mu0}^\pm(g_0 c_0^Z + g_0' c_0^A) - (g_0 Z_{\mu0} + g_0' A_{\mu0})c_0^\pm] \\ \delta_{BRS}Z_{\mu0} &= -\frac{ig_0^2}{\sqrt{g_0^2+g_0'^2}}(W_{\mu0}^+ c_0^- - W_{\mu0}^- c_0^+) + \partial_\mu c_0^Z, \\ \delta_{BRS}A_{\mu0} &= -\frac{ig_0 g_0'}{\sqrt{g_0^2+g_0'^2}}(W_{\mu0}^+ c_0^- - W_{\mu0}^- c_0^+) + \partial_\mu c_0^A, \\ \delta_{BRS}\phi_0 &= -\frac{g_0}{2}(\chi_0^+ c_0^- + \chi_0^- c_0^+) - \frac{\sqrt{g_0^2+g_0'^2}}{2}\chi_{30}c_0^Z, \\ \delta_{BRS}\chi_0^\pm &= +\frac{g_0}{2}[(v_0+\phi_0)c_0^\pm \mp \chi_{30}c_0^\pm)] \pm \frac{i}{2\sqrt{g_0^2+g_0'^2}}\chi_0^\pm[(g_0^2-g_0'^2)c_0^Z + 2g_0 g_0' c^A)], \\ \delta_{BRS}\chi_{30} &= \frac{\sqrt{g_0^2+g_0'^2}}{2}(v_0+\phi_0)c_0^Z - \frac{ig_0}{2}(\chi_0^+ c_0^- - \chi_0^- c_0^+).\end{aligned} \tag{2.82}$$

Here the original ghost fields c_0^3, c_0^0 are replaced by

$$\begin{aligned}c_0^Z &= \frac{1}{\sqrt{g^2+g'^2}}\cdot(g_0 c_0^3 - g_0' c_0^0), \\ c_0^A &= \frac{1}{\sqrt{g^2+g'^2}}\cdot(g_0' c_0^3 + g_0 c_0^0),\end{aligned} \tag{2.83}$$

in the same way as the mixing between physical Z^0 and photon fields. Thus the Faddeev-Popov ghost part, divided into bilinear and cubic terms in fields, is given by

$$\mathcal{L}_{FP} = \mathcal{L}_{FP}^{(2)} + \mathcal{L}_{FP}^{(3)}, \tag{2.84}$$

where

$$\begin{aligned} \mathcal{L}_{FP}^{(2)} &= -\bar{c}_0^+(\partial_\mu^2 + \alpha_{W0}M_{W0}^2)c_0^- - \bar{c}_0^-(\partial_\mu^2 + \alpha_{W0}M_{W0}^2)c_0^+ \\ &\quad -\bar{c}_0^Z(\partial_\mu^2 + \alpha_{Z0}M_{Z0}^2)c_0^Z - \bar{c}_0^A\partial_\mu^2 c_0^A - \bar{c}_0^A(\beta_0 M_{W0}^2)c_0^Z, \end{aligned} \tag{2.85}$$

and

$$\begin{aligned} \mathcal{L}_{FP}^{(3)} &= \frac{ig_0^2}{\sqrt{g_0^2 + g_0'^2}}W_{\mu 0}^+[\partial_\mu \bar{c}_0^- \cdot c_0^Z - \partial_\mu \bar{c}_0^Z \cdot c_0^-] - \frac{ig_0^2}{\sqrt{g_0^2 + g_0'^2}}W_{\mu 0}^-[\partial_\mu \bar{c}_0^+ \cdot c_0^Z - \partial_\mu \bar{c}_0^Z \cdot c_0^+] \\ &\quad +ie_0W_{\mu 0}^+[\partial_\mu \bar{c}_0^- \cdot c_0^A - \partial_\mu \bar{c}_0^A \cdot c_0^-] - ie_0W_{\mu 0}^-[\partial_\mu \bar{c}_0^+ \cdot c_0^A - \partial_\mu \bar{c}_0^A \cdot c_0^+] \\ &\quad +\frac{ig_0^2}{\sqrt{g_0^2 + g_0'^2}}Z_{\mu 0}[\partial_\mu \bar{c}_0^+ \cdot c_0^- - \partial_\mu \bar{c}_0^- \cdot c_0^+] + ie_0A_{\mu 0}^+[\partial_\mu \bar{c}_0^+ \cdot c_0^- - \partial_\mu \bar{c}_0^- \cdot c_0^+] \\ &\quad +i\chi_0^+\left[\frac{-\alpha_{W0}M_{W0}(-g_0'^2 + g_0^2)}{2\sqrt{g_0^2 + g_0'^2}}\bar{c}_0^- c_0^Z - \alpha_{W0}M_{W0}e_0\bar{c}_0^- c_0^A \right. \\ &\qquad \left. +\frac{\beta_0}{2}M_{Z0}g_0\bar{c}_0^A c_0^- + \frac{\alpha_{Z0}}{2}M_{Z0}g_0\bar{c}_0^Z c_0^-\right] \\ &\quad +i\chi_0^-\left[\frac{+\alpha_{W0}M_{W0}(-g_0'^2 + g_0^2)}{2\sqrt{g_0^2 + g_0'^2}}\bar{c}_0^+ c_0^Z + \alpha_{W0}M_{W0}e_0\bar{c}_0^+ c_0^A \right. \\ &\qquad \left. -\frac{\beta_0}{2}M_{Z0}g_0\bar{c}_0^A c_0^+ - \frac{\alpha_{Z0}}{2}M_{Z0}g_0\bar{c}_0^Z c_0^+\right] \\ &\quad +\frac{i\alpha_{W0}}{2}M_{W0}g_0\chi_{30}[-\bar{c}_0^+ c_0^- + \bar{c}_0^- c_0^+] - \frac{\alpha_{W0}}{2}M_{W0}g_0\phi_0[\bar{c}_0^+ c_0^- + \bar{c}_0^- c_0^+] \\ &\quad -\frac{\alpha_{Z0}}{2}M_{Z0}\sqrt{g_0^2 + g_0'^2}\phi_0\bar{c}_0^Z c_0^Z - \frac{\beta_0}{2}M_{Z0}\sqrt{g_0^2 + g_0'^2}\phi_0\bar{c}_0^A c_0^Z. \end{aligned} \tag{2.86}$$

According to our choice of basic parameters, we have to rewrite all the bare constants, g_0, g_0' and v_0, by bare parameters, e_0, M_{Z0} and M_{W0}.

$$\begin{aligned} g_0 &= e_0\frac{M_{Z0}}{\sqrt{M_{Z0}^2 - M_{W0}^2}}, \\ g_0' &= e_0\frac{M_{Z0}}{M_{W0}}, \end{aligned}$$

$$\begin{aligned}
v_0 &= \frac{2}{e_0}\frac{M_{W0}}{M_{Z0}}\sqrt{M_{Z0}^2 - M_{W0}^2}, \\
T_0 &= v_0(\mu_0^2 - \lambda_0 v_0^2)\phi_0 \qquad (2.87) \\
\lambda_0 &= \frac{e_0^2 M_{Z0}^2}{8M_{W0}^2(M_{Z0}^2 - M_{W0}^2)}\left(m_{H0}^2 - \frac{e_0 T_0 M_{Z0}}{M_{W0}\sqrt{M_{Z0}^2 - M_{W0}^2}}\right), \\
\mu_0^2 &= m_{H0}^2/2, \\
f_0^{(f)} &= \sqrt{2}m_{f0}/v_0.
\end{aligned}$$

Here T_0 is the tadpole contribution. After renormalization, it must vanish. By these replacements we can get the final form of the bare Lagrangian.

Renormalization

We have to mention that several kinds of renormalization schemes have been proposed and used so far. The main difference among them lies in the following fact. When one renormalizes the wave functions *before* the symmetry is broken, one puts

$$\begin{aligned}
A_{\mu 0}^a &= Z_W^{1/2} A_\mu^a, \\
B_{\mu 0} &= Z_B^{1/2} B_\mu, \\
\psi_{L0} &= Z_L^{1/2} \psi_L, \qquad (2.88) \\
\psi_{R0}^{(I)} &= Z_R^{(I)1/2} \psi_R^{(I)}, \\
\psi_{R0}^{(i)} &= Z_R^{(i)1/2} \psi_R^{(i)},
\end{aligned}$$

with

$$\psi_L = \begin{pmatrix} \psi_L^{(I)} \\ \psi_L^{(i)} \end{pmatrix}. \qquad (2.89)$$

Hence five constants $Z_W, Z_B, Z_L, Z_R^{(I)}$ and $Z_R^{(i)}$ appear.

On the other hand, *after* the symmetry is broken, one has to introduce more renormalization constants corresponding to fields of physical particles, that is,

$$\begin{aligned}
W_{\mu 0}^\pm &= Z_W^{1/2} W_\mu^\pm, \\
\begin{pmatrix} A_\mu^0 \\ Z_\mu^0 \end{pmatrix} &= \begin{pmatrix} Z_{AA}^{1/2} & Z_{AZ}^{1/2} \\ Z_{ZA}^{1/2} & Z_{ZZ}^{1/2} \end{pmatrix} \begin{pmatrix} A_\mu \\ Z_\mu \end{pmatrix}, \qquad (2.90) \\
\psi_{L0}^{(I)} &= Z_L^{(I)1/2} \psi_L^{(I)}, \\
\psi_{L0}^{(i)} &= Z_L^{(i)1/2} \psi_L^{(i)}, \\
\psi_{R0}^{(I)} &= Z_R^{(I)1/2} \psi_R^{(I)}, \qquad (2.91)
\end{aligned}$$

$$\psi_{R0}^{(i)} = Z_R^{(i)1/2}\psi_R^{(i)},$$

In GRACE system we use the second scheme as our convention.
The Lagrangian is renormalized by the following prescription:

1) Replace the bare constants by

$$\begin{aligned} M_{W0}^2 &= M_W^2 + \delta M_W^2, \\ M_{Z0}^2 &= M_Z^2 + \delta M_Z^2, \\ m_{H0}^2 &= m_H^2 + \delta m_H^2, \\ m_{f0} &= m_f + \delta m_f, \\ e_0 &= Y e. \end{aligned} \tag{2.92}$$

2) Rescale gauge fields$(Z^0, W^\pm, \gamma)$ according to Eq.(2.90).

3) Rescale left- and right-handed fermions by Eq.(2.91). In the presence of quark mixing, Z_L and Z_R become matrices which connect bare and renormalized fermion fields of the *same charge*:

$$\begin{aligned} \psi_{R,L0}^{(f)} &= \sum_{f'} \left(Z_{R,L}^{1/2}\right)_{ff'} \psi_{R,L}^{(f')}, \\ \bar\psi_{R,L0}^{(f)} &= \sum_{f'} \bar\psi_{R,L}^{(f')} \left(Z_{R,L}^{1/2\dagger}\right)_{f'f}. \end{aligned} \tag{2.93}$$

4) Rescale Higgs field by

$$\phi_0 = Z_\phi^{1/2}\phi. \tag{2.94}$$

5) Renormalization of bare gauge parameters appearing in Eq.(2.81), α_{W0}, α_{Z0} and β_0, are defined as follows;

$$\begin{aligned} \alpha_{W0} &= \alpha_W Z_W^{1/2} Z_\chi^{-1/2} / \sqrt{1 + \delta M_W^2/M_W^2}, \\ \alpha_{Z0} &= \alpha_Z Z_{ZZ}^{1/2} Z_{\chi 3}^{-1/2} / \sqrt{1 + \delta M_Z^2/M_Z^2}, \\ \beta_0 &= \alpha_Z Z_{AZ}^{1/2} Z_{\chi 3}^{-1/2} / \sqrt{1 + \delta M_Z^2/M_Z^2}. \end{aligned} \tag{2.95}$$

6) Rescaling of Goldstone fields $\chi_0^\pm, \chi_{30}$ is defined by

$$\begin{aligned} \chi_0^\pm &= Z_\chi^{1/2}\chi^\pm, \\ \chi_{30} &= Z_{\chi 3}^{1/2}\chi_3. \end{aligned} \tag{2.96}$$

7) Rescaling of ghost fields are defined by

$$\begin{aligned}
c_0^\pm &= \tilde{Z}_3 c^\pm, \\
\begin{pmatrix} c_0^Z \\ c_0^A \end{pmatrix} &= \begin{pmatrix} \tilde{Z}_{ZZ}, \tilde{Z}_{ZA} \\ \tilde{Z}_{AZ}, \tilde{Z}_{AA} \end{pmatrix} \begin{pmatrix} c^Z \\ c^A \end{pmatrix}, \\
\bar{c}_0^\pm &= \bar{c}^\pm, \quad \bar{c}_0^Z = \bar{c}^Z, \quad \bar{c}_0^A = \bar{c}^A.
\end{aligned} \tag{2.97}$$

Summary

Now we can re-express the original Lagrangian $\mathcal{L}_{ELW}$ by both renormalized fields and constants. It is straightforward to divide it into free and interaction parts. The free Lagrangian $\mathcal{L}_{free}$ is obtained from the bilinear terms in fields in $\mathcal{L}_{ELW}$ by letting all rescaling factors to be unity, $Z_i = 1$, and all mass counterterms to vanish, $\delta m_i^2 = 0$. Thus we have

$$\begin{aligned}
\mathcal{L}_{free} &= W_\mu^+ \Big[g^{\mu\nu}(\partial_\alpha^2 + M_W^2) - \Big(1 - \frac{1}{\alpha_W}\Big) \partial_\mu \partial_\nu \Big] W_\nu^- \\
&+ \frac{1}{2} Z_\mu \Big[g^{\mu\nu}(\partial_\alpha^2 + M_Z^2) - \Big(1 - \frac{1}{\alpha_Z}\Big) \partial_\mu \partial_\nu \Big] Z_\nu \\
&+ \frac{1}{2} A_\mu \Big[g^{\mu\nu}\partial_\alpha^2 - \Big(1 - \frac{1}{\alpha_A}\Big) \partial_\mu \partial_\nu \Big] A_\nu \\
&+ \sum_f \bar{\psi}^{(f)} (i\partial\!\!\!/ - m_f) \psi^{(f)} - \frac{1}{2} \phi (\partial_\alpha^2 + m_H^2) \phi \\
&- \chi^+ (\partial_\alpha^2 + \alpha_W M_W^2) \chi^- - \frac{1}{2} \chi_3 (\partial_\alpha^2 + \alpha_Z M_Z^2) \chi_3 \\
&- \bar{c}^+ (\partial_\alpha^2 + \alpha_W M_W^2) c^- - \bar{c}^- (\partial_\alpha^2 + \alpha_W M_W^2) c^+ - \bar{c}^A \partial_\alpha^2 c^A \\
&- \bar{c}^Z (\partial_\alpha^2 + \alpha_Z M_Z^2) c^Z
\end{aligned} \tag{2.98}$$

and define the interaction part by

$$\tilde{\mathcal{L}}_{int} \equiv \mathcal{L}_{ELW} - \mathcal{L}_{free}, \tag{2.99}$$

which contains all the counterterms as well as tree interactions. All the Feynman rules generated from the renormalized Lagrangian are collected in chapter 6, except for the counterterms. The latter and the renormalization conditions will be found in Ref.[1]. They are too complicated to be reproduced here.

The Feynman gauge is defined by

$$\alpha_W = \alpha_Z = \alpha_A = 1, \tag{2.100}$$

while the unitary gauge is chosen by letting all these parameters (except for photon) infinity,

$$\alpha_W = \alpha_Z = \infty. \tag{2.101}$$

In GRACE system one can set any values for these parameters when one checks the gauge invariance. In the calculation of cross section, however, *unitary gauge* is automatically chosen, because the total number of diagrams is less than that in general covariant gauge. In the program, parameters with values greater than 100 are regarded to be the unitary gauge (see section 2.4).

2.3.3 QCD

The QCD Lagrangian [2] written in terms of the unrenormalized fields and coupling is given by

$$\begin{aligned}\mathcal{L}_{QCD} &= -\frac{1}{4}G^a_{\mu\nu 0}G_0^{a\mu\nu} + \sum_q \overline{\psi}_0^{(q)}(i\partial\!\!\!/ - m_{q0})\psi_0^{(q)} \\ &\quad + g_0 \sum_q \overline{\psi}_0^{(q)}\gamma^\mu \frac{\lambda_a}{2}\psi_0^{(q)} A^a_{\mu 0} + \mathcal{L}_{gauge} + \mathcal{L}_{ghost}, \qquad (2.102)\end{aligned}$$

where $G^a_{\mu\nu 0}$ is the field strength of gluon defined by

$$G^a_{\mu\nu 0} = \partial_\mu A^a_{\nu 0} - \partial_\nu A^a_{\mu 0} + g_0 f_{abc} A^b_{\mu 0} A^c_{\nu 0}. \qquad (2.103)$$

The color matrix is denoted as $\lambda_a/2, (a = 1, \cdots, 8,$ for $SU(3))$ which satisfies

$$\left[\frac{\lambda_a}{2}, \frac{\lambda_b}{2}\right] = i f_{abc}\frac{\lambda_c}{2}, \qquad \left\{\frac{\lambda_a}{2}, \frac{\lambda_b}{2}\right\} = \frac{1}{3}\delta_{ab} + d_{abc}\frac{\lambda_c}{2}. \qquad (2.104)$$

The constants which appear in the amplitude squared are

$$\begin{aligned}\mathrm{Tr}(1) &= N_c, \qquad \sum_{c,d=1}^{8} f_{acd}f_{bcd} = C_A\delta_{ab}, \\ \mathrm{Tr}\left(\frac{\lambda_a}{2}\frac{\lambda_b}{2}\right) &= T_R\delta_{ab}, \qquad \sum_{a=1}^{8}\frac{\lambda_a}{2}\frac{\lambda_a}{2} = C_F I \qquad (2.105)\end{aligned}$$

where I stands for the unit matrix for the color index. Numerical values of these constants for $SU(3)$ are

$$N_c = 3, \quad C_A = 3, \quad C_F = \frac{4}{3}, \quad T_R = \frac{1}{2}. \qquad (2.106)$$

The last term $\mathcal{L}_{ghost}$ in Eq.(2.102) is for the ghost particle. There are two gauges widely used. These are the same as in QED but have the following features:

1. Covariant gauge:
   Ghost particles should be introduced. These particles are needed whenever gluon loop is formed, even in the final state of the squared amplitude. Gauge parameter α appears in Eq.(2.50) for the covariant gauge is denoted to α_G.

2. Axial gauge:
   No unphysical ghost particle is needed.

The gauge fixing term $\mathcal{L}_{gauge}$ is exactly the same as in QED (see Eq.(2.43)). `GRACE` allows to choose either of these gauges. See section 3.2.1.

The renormalization can be done in the same way as in QED. The rescaling factors for unrenormalized quantities are introduced by

$$\begin{aligned} A^a_{\mu 0} &= Z_3^{1/2} A^a_\mu, \qquad \psi_0^{(q)} = Z_{2q}^{1/2}\psi^{(q)}, \\ g_0 &= Z_g g \\ m_{q0} &= Z_{mq} m_q. \end{aligned} \tag{2.107}$$

Quark and gluon renormalization constants are denoted as Z_{2q} and Z_3, resectively and that for the strong coupling is Z_g. Quark mass is renormalized in the multiplicative way. We substitute these relations into $\mathcal{L}_{QCD}$ to eliminate bare quantities and after that we separate counterterms from the rest,

$$\begin{aligned} \mathcal{L}_{QCD} &= -\frac{1}{4}F^a_{\mu\nu}F^a_{\mu\nu} + \sum_q \overline{\psi}^{(q)}(i\partial\!\!\!/ - m_q)\psi^{(q)} + g\sum_q \overline{\psi}^{(q)}\gamma_\mu \frac{\lambda_a}{2}\psi^{(q)}A^a_\mu \\ &\quad -g f_{abc}(\partial_\mu A^a_\nu)A^b_\mu A^c_\nu - \frac{1}{4}g^2 f_{abc}f_{ade}A^b_\mu A^c_\nu A^d_\mu A^e_\nu \\ &\quad +\delta\mathcal{L}_c + \mathcal{L}_{gauge} + \mathcal{L}_{ghost}, \\ &= \mathcal{L}_{free} + \delta\mathcal{L}_c + \mathcal{L}_{int}, \end{aligned} \tag{2.108}$$

Here $\mathcal{L}_{free}$ is the bilinear form of fields including gauge fixing and ghost terms. The tensor $F^a_{\mu\nu}$ is introduced to express the linear part of gluon field,

$$F^a_{\mu\nu} = \partial_\mu A^a_\nu - \partial_\nu A^a_\mu. \tag{2.109}$$

The counterterm Lagrangian $\delta\mathcal{L}_c$ is given by

$$\begin{aligned} \delta\mathcal{L}_c &= -\frac{1}{4}\delta Z_3 F^a_{\mu\nu}F^a_{\mu\nu} \\ &\quad +\sum_q \delta Z_{2q}\overline{\psi}^{(q)}(i\partial\!\!\!/ - m_q)\psi^{(q)} - \sum_q \delta m_q \overline{\psi}^{(q)}\psi^{(q)} + g\sum_q \delta Z_{fq}\overline{\psi}^{(q)}\gamma_\mu\frac{\lambda_a}{2}\psi^{(q)}A^a_\mu \\ &\quad -g\delta Z_t f_{abc}(\partial_\mu A^a_\nu)A^b_\mu A^c_\nu - \frac{1}{4}g^2\delta Z_q f_{abc}f_{ade}A^b_\mu A^c_\nu A^d_\mu A^e_\nu \\ &\quad +\{\text{counterterm for ghost fields}\}. \end{aligned} \tag{2.110}$$

Constants in the counterterms are defined by

$$Z_{2q} = 1 + \delta Z_{2q},$$

$$\begin{aligned} Z_3 &= 1+\delta Z_3, \\ Z_g Z_{2q} Z_3^{1/2} &= 1+\delta Z_{fq}, \\ Z_g Z_3^{3/2} &= 1+\delta Z_t, \\ Z_g^2 Z_3^2 &= 1+\delta Z_q, \\ \delta m_q &= (Z_{2q} Z_{mq} - 1) m_q. \end{aligned} \tag{2.111}$$

Renormalization constants of interaction vertices, Z_f, Z_t and Z_q, correspond to quark-quark-gluon(qqG), triple-gluon(GGG) and quadruple-gluon($GGGG$) coupling, respectively. Feynman rules are given in chapter 6. Counterterms will be found in Ref.[2], which we do not write down here. In massless QCD as the coupling constant g can be renormalized at an arbitrary mass scale μ,

$$g_0 = Z_g(\mu) g(\mu), \tag{2.112}$$

the running coupling constant $g(\mu)$ is determined from the β-function defined by

$$\mu \frac{\mathrm{d}g(\mu)}{\mathrm{d}\mu} = \beta(g(\mu)). \tag{2.113}$$

This β-function is expanded up to the second order as

$$\beta(g) = -\frac{g^3}{(4\pi)^2}\beta_0 - \frac{g^5}{(4\pi)^4}\beta_1 + O(g^7). \tag{2.114}$$

Then the lowest order running coupling constant $\alpha_S(Q^2) \equiv g^2(Q^2)/4\pi$ is given by

$$\alpha_S^{(0)}(Q^2) = \frac{4\pi}{\beta_0 \cdot \ln(Q^2/\Lambda_{QCD}^2)}, \tag{2.115}$$

where the parameter Λ_{QCD} is defined by

$$1 + \frac{\alpha_S^{(0)}(\mu^2)}{4\pi} \cdot \beta_0 \cdot \ln\left(\frac{\Lambda_{QCD}^2}{\mu^2}\right) = 0, \tag{2.116}$$

with

$$\beta_0 = \frac{11C_A - 4T_R N_F}{3}, \tag{2.117}$$

where N_F is the number of flavors. Including the next order term in $\beta(g)$, we find the coupling constant in the next-to-leading-logarithmic approximation

$$\frac{1}{\alpha_S^{(1)}(Q^2)} + \frac{\beta_1}{4\pi\beta_0}\ln\Big(\frac{\alpha_S^{(1)}(Q^2)}{4\pi\beta_0/\beta_1 + \alpha_S^{(1)}(Q^2)}\Big) = \frac{\beta_0}{4\pi}\ln(\frac{Q^2}{\Lambda^2}), \tag{2.118}$$

which can be approximated by the following expansion formula

$$\alpha_S^{(1)}(Q^2) \simeq \alpha_S^{(0)}(Q^2)\left[1 - \frac{\beta_1}{\beta_0^2}\frac{\ln\ln(Q^2/\Lambda^2)}{\ln(Q^2/\Lambda^2)}\right]. \tag{2.119}$$

where

$$\beta_1 = \frac{34C_A^2 - (20C_A + 12C_F)T_R N_F}{3}. \tag{2.120}$$

In `GRACE` system, contrary to the case of QED or electroweak theory, *the strong running coupling constant α_S is not generated.* The reason is because it cannot be uniquely defined but depends on a momentum characteristic to the process and thus may differ from one process to another. The user should define it in the subroutine `KINEM` and include it in the variable `YACOB` which is multiplied to the amplitude squared. In this way one can introduce the coupling constant of any variable, such as $\alpha_S(s)$ or $\alpha_S(p_T^2)$ and so on, most suitable one to the problem.

2.4 Method of amplitude calculation

The procedure of the numerical calculation of Feynman amplitudes is divided into the following steps: Amplitudes are decomposed to vertex amplitudes by splitting internal lines into wave functions of fermions and polarization vectors of vector bosons. Using the decompositions of the internal lines we can write any Feynman amplitudes in terms of vertex amplitudes. Then the vertex amplitudes are numerically calculated. Finally the internal lines are reproduced numerically by summing over spin states for the fermions and polarization states for the vector bosons, respectively.

This method enables us to construct compact programs for calculation of Feynman amplitude in any tree Feynman diagrams for the electroweak theory. Taking into account the color factors, we can also use this method for QCD(see section 2.4.3). Corresponding program package `CHANEL` is presented in section 7.3.

2.4.1 Calculation of amplitudes

Let us consider a scattering amplitude corresponding to the Feynman graph shown in figure 2.1 as an example.

Fig. 2.1 A Feynman graph for the process $e^+e^- \to W^+W^-\gamma$.

In this graph p_1, p_2, q_1 q_2 and k are momenta of e^+, e^-, W^+, W^- and γ, and h_1 and h_2 are helicities of e^+ and e^-, and $\epsilon_1(q_1)$, $\epsilon_2(q_2)$ and $\epsilon_3(k)$ are polarization vectors of W^+, W^- and γ, respectively.

The scattering amplitude for this graph is given by

$$\begin{aligned} T_{fi} &= \overline{v}(p_1, h_1)\, c^{\eta}_{eW}\, \epsilon_{1\eta}(q_1)\, S_F(-p_1+q_1)\, c^{\mu}_{eW}\, u(p_2, h_2) \\ &\quad \times D_{F\,\mu\nu}(q_2+k)\, c^{\nu\rho\sigma}_{WW\gamma}(q_2+k, -q_2, -k)\, \epsilon_{2\rho}(q_2)\, \epsilon_{3\sigma}(k), \end{aligned} \tag{2.121}$$

where c^{η}_{eW} and $c^{\nu\rho\sigma}_{WW\gamma}$ express electron-W and photon-W couplings, respectively, and they are given by:

$$c^{\eta}_{eW} = \frac{eM_Z}{\sqrt{2(M_Z^2 - M_W^2)}}\gamma^{\eta}\,\frac{1-\gamma_5}{2} \tag{2.122}$$

and

$$c^{\nu\rho\sigma}_{WW\gamma}(p,q,r) = e[(p-q)^{\sigma}g^{\nu\rho} + (q-r)^{\nu}g^{\rho\sigma} + (r-p)^{\rho}g^{\sigma\nu}]. \tag{2.123}$$

The key observation used in CHANEL is that propagators can be expressed by bilinear form of wave functions:

$$S_F(p) = \frac{\sum_{\alpha,i} w_{\alpha,i}\, U^{\alpha}(h^{(i)},p^{(i)})\,\overline{U}^{\alpha}(h^{(i)},p^{(i)})}{p^2 - m^2} \tag{2.124}$$

and

$$D_{F\,\mu\nu}(p) = \frac{\sum_i\, w_i\, \epsilon^{(i)}_{\mu}(p)\,\epsilon^{(i)}_{\nu}(p)}{p^2-m^2}, \tag{2.125}$$

where $w_{\alpha,j}$ and w_i are c-numbers, weight factors for the decomposition of propagator, and U^{α} represents either spinor u or anti-spinor v depending on the value of index α. Momenta $p^{(i)}$ are calculated from off-shell fermion momentum p.

By substituting these expressions, we obtain

$$\begin{aligned}
T_{fi} &= \frac{1}{(-p_1+q_1)^2}\,\frac{1}{(q_2+k)^2 - M_W^2}\sum_{\alpha,i} w_{\alpha,i}\sum_{l} w_l \\
&\quad \times\overline{v}(p_1,h_1)\,c^{\eta}_{eW}\,\epsilon_{1\eta}(q_1)\,U^{\alpha}(h^{(i)},(-p_1+q_1)^{(i)}) \\
&\quad \times\overline{U}^{\alpha}(h^{(i)},(-p_1+q_1)^{(i)})\,c^{\mu}_{eW}\,\epsilon^{(l)}_{\mu}(q_2+k)\,u(p_2,h_2) \\
&\quad \times c^{\nu\rho\sigma}_{WW\gamma}(q_2+k,-q_2,-k)\,\epsilon^{(l)}_{\nu}(q_2+k)\,\epsilon_2(q_2)_{\rho}\,\epsilon_3(k)_{\sigma} \qquad (2.126)\\
&= \frac{1}{D(-p_1+q_1,0)}\frac{1}{D(q_2+k,M_W)}\sum_{\alpha,i} w_{\alpha,i}\sum_{l} w_l \\
&\quad \times V^{(\alpha,i)}_{eW^+}\,V^{(\alpha,i,l)}_{eW^-}\,V^{(l)}_{WW\gamma},
\end{aligned}$$

where

$$\begin{aligned}
D(p,m) &= p^2 - m^2, \\
V^{(\alpha,i)}_{eW^+} &= \overline{v}(p_1,h_1)\,c^{\eta}_{eW}\,\epsilon_{1\eta}(q_1)\,U^{\alpha}(h^{(i)},(-p_1+q_1)^{(i)}), \qquad (2.127)\\
V^{(\alpha,i,l)}_{eW^-} &= \overline{U}^{\alpha}(h^{(i)},(-p_1+q_1)^{(i)})\,c^{\mu}_{eW}\,\epsilon^{(l)}_{\mu}(q_2+k)\,u(p_2,h_2),
\end{aligned}$$

and

$$V^{(l)}_{WW\gamma} = c^{\nu\rho\sigma}_{WW\gamma}(q_2+k,-q_2,-k)\,\epsilon^{(l)}_{\nu}(q_2+k)\,\epsilon_{2\rho}(q_2)\,\epsilon_{3\sigma}(k). \tag{2.128}$$

Calculation of the vertex parts $V^{(\alpha,i)}_{eW^+}$, $V^{(\alpha,i)}_{eW^-}$ and $V^{(l)}_{WW\gamma}$ are prepared as subroutines in the program library CHANEL. Numerical value of the amplitude is easily obtained by them.

2.4.2 Formulas for amplitude calculations

In this section, we present the basic formulas for the amplitude calculations. We shall start to derive vertex amplitude for the fermion-fermion-vector boson (FFV) vertex. Here we define helicity eigenstates of massless fermion (not antifermion) with momentum p as

$$\chi_\lambda(p) = \not{p}\chi_{-\lambda}(k_0)/\sqrt{2(p\cdot k_0)}, \tag{2.129}$$

where $\lambda = \pm$ denote helicity states for fermion and k_0 is a reference momentum to be specified. The basic spinor $\chi_\lambda(k_0)$ is defined such that the helicity state $\chi_\lambda(k_0)$ satisfies usual relation to chirality projection operator:

$$\chi_\pm(k_0)\bar{\chi}_\pm(k_0) = \omega_\pm \not{k}_0 \tag{2.130}$$

where $\omega_\pm$ is defined as

$$\omega_\pm = (1 \pm \gamma_5)/2. \tag{2.131}$$

Here the positive-helicity state is fixed by the relation

$$\chi_+(k_0) = \not{k}_1\chi_-(k_0), \tag{2.132}$$

where k_1 is chosen in such a way as $k_1^2 = -1$ and $k_1\cdot k_0 = 0$, which guarantee that $\chi_+(k_0)$ satisfies Eq.(2.130). Due to these conditions, the wave function defined in Eq.(2.129) also satisfies the relation

$$\chi_\pm(p)\bar{\chi}_\pm(p) = \omega_\pm\not{p}. \tag{2.133}$$

Since the helicity state of antifermion has opposite sign to that of fermion in the same chirality state, we define the wave function for massless fermion and that for antifermion with helicity h as

$$\chi^\rho(h,p) \equiv \chi_{\rho h}(p), \tag{2.134}$$

where $\rho = +$ for fermion and $\rho = -$ for antifermion, respectively. Using Eqs.(2.129) to (2.131), we can write vertex amplitudes for massless fermions in terms of components of momenta and polarization vector of the vector boson ϵ,

$$\bar{\chi}^{\rho'}(h',p')\not{\epsilon}(q)\Gamma\chi^\rho(h,p) = \delta_{\rho'h',\rho h}A_{\rho h}(R_1 + i\rho h R_2), \tag{2.135}$$

with

$$\Gamma = A_+\omega_+ + A_-\omega_-, \tag{2.136}$$

where

$$R_1 = \{(k_0\cdot p')(\epsilon\cdot p) - (k_0\cdot\epsilon)(p'\cdot p) + (k_0\cdot p)(p'\cdot\epsilon)\}/\sqrt{(p'\cdot k_0)(p\cdot k_0)}$$

and

$$R_2 = \varepsilon_{\mu\nu\rho\sigma}k_0^\mu\epsilon^\nu p'^\rho p^\sigma/\sqrt{(p'\cdot k_0)(p\cdot k_0)}. \tag{2.137}$$

In Eq.(2.136), $A_\pm$ denote coupling constants for left ($-$) and right handed ($+$) chirality of vertices.

In actual computations, it is convenient to specify k_0 such that the forms of R_1 and R_2 become compact. For instance, choosing $k_0 = (1,1,0,0)$, we can write Eq.(2.137) as

$$\begin{aligned} R_1 &= (\epsilon^0 + \epsilon^x)\sqrt{\tilde{p}'^0\tilde{p}^0} - (\hat{\epsilon}\cdot\hat{r})\sqrt{\tilde{p}'^0} - (\hat{\epsilon}\cdot\hat{r}')\sqrt{\tilde{p}^0} + (\epsilon^0 - \epsilon^x)(\hat{r}'\cdot\hat{r}), \\ R_2 &= (\boldsymbol{\epsilon}\times\boldsymbol{r})^x\sqrt{\tilde{p}'^0} - (\boldsymbol{\epsilon}\times\boldsymbol{r}')^x\sqrt{\tilde{p}^0} - (\epsilon^0 - \epsilon^x)(\boldsymbol{r}'\times\boldsymbol{r})^x, \end{aligned} \tag{2.138}$$

where $\tilde{p}^0 = p^0 - p^x$ and $\hat{r} = \hat{p}/\sqrt{\tilde{p}^0}$. Here $\hat{\epsilon}$ and $\hat{r}$ are $\hat{\epsilon} = (\epsilon^y, \epsilon^z)$ and $\hat{r} = (r^y, r^z)$, respectively.

Next we extend these expressions to the case of massive fermions. We define a wave function of massive fermion as

$$U^\rho(h,p,m) = c(\not{p} + \rho m)\chi_{-\rho h}(k)/\sqrt{2(p\cdot k)}, \tag{2.139}$$

where an light-like vector k is fixed so that the U^ρ is an eigenstate of helicity h. Moreover the complex phase c is chosen such that U^ρ becomes χ^ρ defined in Eq.(2.129) at the limit where the fermion mass goes to zero. Notice that redefinition of the overall phase of U^ρ does not affect final results since amplitudes are squared in cross sections. In order to satisfy above conditions we choose two light-like vectors which build up p as

$$p = p_1 + p_2,$$

with

$$p_1 = \frac{(p^0 + |\boldsymbol{p}|)}{2}(n + n_p),$$

and

$$p_2 = \frac{m^2}{2(p^0 + |\boldsymbol{p}|)}(n - n_p), \tag{2.140}$$

where $n = (1,0,0,0)$ and $n_p = (0, p^x/|\boldsymbol{p}|, p^y/|\boldsymbol{p}|, p^z/|\boldsymbol{p}|)$. Under this decomposition of the momentum p and choosing $k = p_2$,we can verify that U^ρ satisfies the relation

$$U^\rho(h,p,m)\bar{U}^\rho(h,p,m) = (1 + h\gamma_5\not{s})(\not{p} + \rho m)/2, \tag{2.141}$$

where

$$s = \frac{|\boldsymbol{p}|}{m}n + \frac{p^0}{m}n_p. \tag{2.142}$$

Notice that s obtained in Eq.(2.142) is a helicity axis of fermion.
Furthermore, Eq.(2.139) is written by the two wave functions for massless fermions $\chi_\pm$:

$$U^\rho(h,p,m) = \chi_{\rho h}(p_1) - \rho c_{-\rho h}(p)\chi_{-\rho h}(p_2), \tag{2.143}$$

where

$$c_+(p) = \frac{(k_0\cdot n_p)(k_1\cdot n) - (k_0\cdot n)(k_1\cdot n_p) + i\varepsilon_{\mu\nu\rho\sigma}k_0^\mu k_1^\nu n^\rho n_p^\sigma}{\sqrt{(k_0\cdot n)^2 - (k_0\cdot n_p)^2}} \tag{2.144}$$

and

$$c_-(p) = -{c_+}^*(p). \tag{2.145}$$

For $k_0 = (1,1,0,0)$ and $k_1 = (0,0,1,0)$, the phase factor for the wave function of massive fermion in Eq.(2.144) is reduced to

$$c_+(p) = \frac{p^y + ip^z}{\sqrt{(p^y)^2 + (p^z)^2}}. \tag{2.146}$$

The general form presented in Eq.(2.143) is convenient to construct the vertex amplitudes of fermion-fermion-vector boson vertex for massive fermions. Here we write a general form of vertex amplitude as

$$\begin{aligned} J^{[V]\rho'\rho}_{h'h\lambda}(m', m, p', p, q, A_-, A_+) &= \bar{U}^{\rho'}(h', p', m')\not{\epsilon}_\lambda(q)\Gamma U^\rho(h, p, m) \\ &\equiv J^{[V]}_{\rho'h',\rho h,\lambda}(m', m, p', p, q, A_-, A_+), \end{aligned} \tag{2.147}$$

where Γ is defined in Eq.(2.136). Using Eq.(2.143), we can decompose the vertex amplitudes for massive fermions in terms of linear combinations of those for fermion-fermion-vector (FFV) boson vertices for massless fermions obtained in Eq.(2.135):

$$\begin{aligned} J^{[V]}_{\pm,\pm,\lambda}(m', m, p', p, q, A_-, A_+) &= A_\pm \bar{\chi}_\pm({p'}_1)\not{\epsilon}_\lambda(q)\chi_\pm(p_1) \\ &\quad + A_\mp \rho'\rho {c^*}_\mp(p')c_\mp(p)\bar{\chi}_\mp(p'_2)\not{\epsilon}_\lambda(q)\chi_\mp(p_2), \\ J^{[V]}_{\pm,\mp,\lambda}(m', m, p', p, q, A_-, A_+) &= -A_\pm \rho c_\pm(p)\bar{\chi}_\pm(p'_1)\not{\epsilon}_\lambda(q)\chi_\pm(p_2) \\ &\quad - A_\mp \rho' {c^*}_\mp(p')\bar{\chi}_\mp({p'}_2)\not{\epsilon}_\lambda(q)\chi_\mp(p_1), \end{aligned} \tag{2.148}$$

where momenta p and p' are decomposed by Eq.(2.140).

Vertex amplitude for fermion-fermion-scalar boson (FFS) vertex is also written by a similar form;

$$\begin{aligned} J^{[S]\rho'\rho}_{h'h}(m', m, p', p, A_-, A_+) &= \bar{U}^{\rho'}(h', p', m')\Gamma U^\rho(h, p, m) \\ &\equiv J^{[S]}_{\rho'h',\rho h}(m', m, p', p, A_-, A_+). \end{aligned} \tag{2.149}$$

We can decompose the vertex amplitudes for massive fermions in terms of linear combinations of FFS vertices for massless fermions;

$$\begin{aligned} J^{[S]}_{\pm,\mp}(m', m, p', p, A_-, A_+) &= A_\mp \bar{\chi}_\pm({p'}_1)\chi_\mp(p_1) \\ &\quad + A_\pm \rho'\rho {c^*}_\mp(p')c_\pm(p)\bar{\chi}_\mp(p'_2)\chi_\pm(p_2), \\ J^{[S]}_{\pm,\pm}(m', m, p', p, A_-, A_+) &= -A_\mp \rho c_\mp(p)\bar{\chi}_\pm(p'_1)\chi_\mp(p_2) \\ &\quad - A_\pm \rho' {c^*}_\mp(p')\bar{\chi}_\mp({p'}_2)\chi_\pm(p_1). \end{aligned} \tag{2.150}$$

The expressions of the vertex amplitude for massless fermion-scalar boson vertex are written as

$$\bar{\chi}^{\rho'}(h',p')\Gamma\chi^{\rho}(h,p) = \delta_{-\rho' h',\rho h}A_{\rho h}(R_1 + i\rho h R_2) \tag{2.151}$$

with

$$\Gamma = A_+\omega_+ + A_-\omega_-, \tag{2.152}$$

where

$$\begin{aligned} R_1 &= \frac{(k_0\cdot p')(k_1\cdot p) - (k_0\cdot p)(k_1\cdot p')}{\sqrt{(p'\cdot k_0)(p\cdot k_0)}}, \\ R_2 &= \frac{\varepsilon_{\mu\nu\rho\sigma}k_0^{\mu}k_1^{\nu}p^{\rho}p'^{\sigma}}{\sqrt{(p'\cdot k_0)(p\cdot k_0)}}. \end{aligned} \tag{2.153}$$

Choosing $k_0 = (1,1,0,0)$ and $k_1 = (0,0,1,0)$, we can write Eq.(2.153) as

$$\begin{aligned} R_1 &= p'^y\sqrt{\frac{\tilde{p}^0}{\tilde{p}'^0}} - p^y\sqrt{\frac{\tilde{p}'^0}{\tilde{p}^0}} \\ R_2 &= p'^z\sqrt{\frac{\tilde{p}^0}{\tilde{p}'^0}} - p^z\sqrt{\frac{\tilde{p}'^0}{\tilde{p}^0}}, \end{aligned} \tag{2.154}$$

with $\tilde{p}^0 = p^0 - p^x$.

Notice that combinations of helicity states for the FFS vertex are different from those of FFV vertex.

The vertex amplitude for three vector boson vertex is given by

$$\begin{aligned} V^{[3]}_{\lambda_1,\lambda_2,\lambda_3}(q_1,q_2,q_3) &= G_{VVV}[((q_1-q_2)\cdot\epsilon_{\lambda_3}(q_3))(\epsilon_{\lambda_1}(q_1)\cdot\epsilon_{\lambda_2}(q_2)) \\ &\quad +((q_2-q_3)\cdot\epsilon_{\lambda_1}(q_1))(\epsilon_{\lambda_2}(q_2)\cdot\epsilon_{\lambda_3}(q_3)) \\ &\quad +((q_3-q_1)\cdot\epsilon_{\lambda_2}(q_2))(\epsilon_{\lambda_3}(q_3)\cdot\epsilon_{\lambda_1}(q_1))], \end{aligned} \tag{2.155}$$

where G_{VVV} is the coupling constant of the self-interactions.

Four point vertex of vector bosons is given by the following simple form:

$$\begin{aligned} V^{[4]}_{\lambda_1,\lambda_2,\lambda_3,\lambda_4}(q_1,q_2,q_3,q_4) &= G_{VVVV}[(\epsilon_{\lambda_1}(q_1)\cdot\epsilon_{\lambda_3}(q_3))(\epsilon_{\lambda_2}(q_2)\cdot\epsilon_{\lambda_4}(q_4)) \\ &\quad +(\epsilon_{\lambda_1}(q_1)\cdot\epsilon_{\lambda_4}(q_4))(\epsilon_{\lambda_2}(q_2)\cdot\epsilon_{\lambda_3}(q_3)) \\ &\quad -2(\epsilon_{\lambda_1}(q_1)\cdot\epsilon_{\lambda_2}(q_2))(\epsilon_{\lambda_3}(q_3)\cdot\epsilon_{\lambda_4}(q_4))]. \end{aligned} \tag{2.156}$$

Finally scalar and vector boson vertices are written in the following forms: For scalar-scalar-vector boson (SSV) vertex,

$$J^{[SSV]}_{\lambda}(q,p_1,p_2) = G_{SSV}[\epsilon_{\lambda}(q)\cdot(p_1-p_2)] \tag{2.157}$$

and for VVS and $VVSS$ vertices,

$$V^{[VVS(S)]}_{\lambda_1,\lambda_2}(q_1,q_2) = G_{VVS(S)}(\epsilon_{\lambda_1}(q_1)\cdot\epsilon_{\lambda_2}(q_2)). \tag{2.158}$$

The coupling constants should be assigned in a consistent way in order to give correct relative signs among amplitudes.

Using above expressions for vertices, we can numerically calculate the vertex amplitudes included in the amplitude of any tree graph in electroweak theories.

Next we write a numerator of internal fermion line $\not{q} + m$ in terms of wave functions of on-shell fermion fields. Here we decompose the momentum q in terms of a light-like vector $\tilde{l}_1$ and a time-like vector $\tilde{l}_2$ with the square of the four vector m^2. Using the relation of Eq.(2.141) ,we can write

$$\not{q} + m = \sum_{h=\pm} \{\mathrm{sign}(\tilde{l}_1^0) U^+(h, l_1, 0) \bar{U}^+(h, l_1, 0) + \mathrm{sign}(\tilde{l}_2^0) U^{\rho_2}(h, l_2, m) \bar{U}^{\rho_2}(h, l_2, m)\}, \tag{2.159}$$

where $\mathrm{sign}(\tilde{l}_i^0)$ denotes the sign of the time component of vector $\tilde{l}_i$, and l_i is defined by

$$\tilde{l}_i = \mathrm{sign}(\tilde{l}_i^0) l_i. \tag{2.160}$$

Here ρ_2 is chosen as $\rho_2 = \mathrm{sign}(\tilde{l}_2^0)$.

Numerator of propagator for massive vector boson in covariant gauge is written as

$$G_{\mu\nu}(q) = -g^{\mu\nu} + (1 - \alpha) \frac{q^\mu q^\nu}{q^2 - \alpha M^2}, \tag{2.161}$$

where M and α denote the mass of vector boson and the gauge parameter, respectively. $G_{\mu\nu}(q)$ is also decomposed by polarization vectors as

$$G_{\mu\nu}(q) = \sum_{\lambda} \epsilon_{\lambda\mu}(q) \epsilon_{\lambda\nu}(q) \eta_\lambda(q). \tag{2.162}$$

The expressions of polarization vectors ϵ_λ^μ depend on the gauge choice. For instance, we choose as a rectangular polarization basis,

$$\begin{aligned}
\epsilon^\mu_{\lambda=1}(q) &= \frac{1}{q_T |\boldsymbol{q}|}(0, q^x q^z, q^y q^z, -q_T^2), \\
\epsilon^\mu_{\lambda=2}(q) &= \frac{1}{q_T}(0, -q^y, q^x, 0), \\
\epsilon^\mu_{\lambda=3}(q) &= \frac{q^0}{Q|\boldsymbol{q}|}(\frac{|\boldsymbol{q}|^2}{q^0}, q^x, q^y, q^z), \\
\epsilon^\mu_{\lambda=4}(q) &= \frac{q^\mu}{Q},
\end{aligned} \tag{2.163}$$

with $Q = \sqrt{|q^2|}$ and $q_T^2 = (q^x)^2 + (q^y)^2$. The polarization vectors with $\lambda = 1, 2$ correspond to the transverse components and $\lambda = 3$ denotes the longitudinal one. The polarization vector with $\lambda = 4$ should be added if massive vector bosons become virtual states.

Using a rectangular polarization basis presented in Eq.(2.163), $G_{\mu\nu}(q)$ can be reproduced by choosing the weight factor η_λ as follows:

$$\begin{aligned}\eta_1 &= \eta_2 = +1 \\ \eta_3 &= \text{sign}(q^2) \\ \eta_4 &= \text{sign}(q^2)\frac{\alpha(M^2 - q^2)}{q^2 - \alpha M^2},\end{aligned} \tag{2.164}$$

where sign(q^2) means $+1$ for $q^2 > 0$ and -1 for $q^2 < 0$, respectively.

The unitary gauge is chosen for $\alpha \geq 100$ with $M > 0$, since the unitary gauge corresponds to $\alpha \to \infty$, which is not appropriate for numerical calculations.

By varying the gauge parameters, we can check the gauge invariance of calculated amplitudes when they contain vector boson propagators.

Using Eqs.(2.159) and (2.162), we can write any Feynman amplitudes in terms of vertex amplitudes.

According to above expressions, we can numerically calculate any Feynman amplitudes for the electroweak theory in tree level. Although the program package for the numerical calculations are presented in section 7.3, we shall briefly explain the relation between the program package and the obtained expressions.

The program package `CHANEL` consists of subroutines to calculate vertex amplitudes (`FFV`, `FFVO`, `FFS`, `FFSO`, `VVV`, `VVVV`, `VVS`, `VVSS` and `SSV`). `CHANEL` also contains the subroutines to obtain the polarization vectors and the weight factors of the vector boson (`POLA`), phase factors of the wave function of fermion (`PHASEQ`) and the subroutines to decompose the momentum of fermion (`SPLT` and `SPLTQ`).

2.4.3 Color factor

The one of basic principle of the `GRACE` system is to calculate the amplitude rather than its squared form. Though the color is a freedom of particles like helicity, we treat the color in a different way. If we consider the color factor in the amplitude level, we must sum and/or average for all color states after squaring the amplitudes in the initial and final states as we do for helicity states. While it is interesting to calculate the amplitude for some fixed helicity states, it is practically meaningless to do for some fixed color states. Further, the color factor is independent of momenta. Therefore, it is simpler to treat the color as a factor multiplied to each squared matrix element than to handle it in amplitudes. Hence we include the color factor when we calculate the square of the sum of amplitudes for the process in such a way:

$$\sum_{i,j=1}^{k} C_{ij} M_i M_j^\dagger, \tag{2.165}$$

where k is the number of amplitude, M_i is the i-th amplitude in which the color factors are removed (we do not use T as amplitude here), and C_{ij} is the color factor for the matrix element $M_i M_j^\dagger$. In this subsection we present the algorithm to calculate C_{ij}. We consider the group $SU(N_c)$ for the color degree of freedom and explicit implementation is done for $N_c = 3$. The fundamental representation is given by T^a ($= \lambda_a/2$ in section 2.3.3) and it satisfies

$$[T^a, T^b] = i f_{abc} T^c, \tag{2.166}$$

where f_{abc} is the structure constants of $SU(N_c)$. The following relations hold(see section 2.3.3. $T_a = \lambda_a/2$):

$$\begin{aligned} T^a T^a &= C_F \cdot I = \frac{N_c^2 - 1}{2N_c} \cdot I, \\ \mathrm{Tr}(T^a T^b) &= T_R \delta^{ab} = \frac{1}{2}\delta^{ab}, \\ f_{abc} f_{dbc} &= C_A \delta^{ad} = N_c \delta^{ad}, \end{aligned} \tag{2.167}$$

(repeated indices are summed).
The color charge is carried only by quarks, gluons and ghost particles for gluons. We follow the notation and convention of Ref.[2]. In the calculation of color factor, particles without color charge are neglected. Denoting the QCD coupling constant as g, we have three types of vertices:

(1) quark-gluon vertex	gT^a
(2) three-gluon vertex	$-igf_{abc}$
(3) four-gluon vertex	$-g^2(f_{abe}f_{cde}+$ cyclic permutation).

Here we only show the color factor (see Eq.(2.109)) and the ghost-gluon vertex is the same as the three-gluon vertex as far as the color factor is concerned.

First we replace the four-gluon vertex by the sum of three diagrams (of s, t, u-types) with three three-gluon vertices:

(2.168)

$(-ig)f_{abx}(-ig)f_{cdy}$ $(-ig)f_{dax}(-ig)f_{bcy}$ $(-ig)f_{cax}(-ig)f_{dby}$

Here the dashed line represents a gluon color propagator, δ_{xy}, which has neither Lorentz structure nor momentum dependence.

In the second step, we replace the three-gluon vertex by the sum of two quark loops:

$$g\mathrm{Tr}(T^aT^bT^c) \qquad\qquad g\mathrm{Tr}(T^bT^aT^c)$$

Here, the identity

$$-\,igf_{abc} = -g\mathrm{Tr}[T^a, T^b]T^c \tag{2.169}$$

is used.

After the last step, the diagram becomes QED type. We apply the Fierz transformation for color,

$$(T^a)_{ij}(T^a)_{kl} = -(1/2N_c)\delta_{ij}\delta_{kl} + (1/2)\delta_{il}\delta_{jk} \tag{2.170}$$

on each gluon propagator to express it as two pairs of quark lines.

Finally we are left with graphs which consists of only quarks. In summary, the color factor of a diagram for a matrix element with n_4 four-gluon vertices, n_3 three-gluon vertices, and n_g gluon propagator is equivalent to $3^{n_4}2^{n_3}2^{n_g+n_4}$ graphs which consist of only quark lines. The color factor of each graph is $3^n \cdot C$ where n is the number of quark loops and C is the product of factors at each decomposition given in Eqs.(2.168), (2.169), and (2.170).

Further, we need average for color states if some of colored particles belong to the initial state. For a quark in the initial state, we divide by N_c and for a gluon by $N_c^2 - 1$.

2.5 Feynman graph generation

Graph generation subsystem is designed by the following guiding principles:

1) The current version of GRACE system supports amplitude calculation only for the tree level. Since, however, it will be extended to inclusion of one-loop corrections in near future, the current graph generation subsystem is desirable to be able to generate one-loop as well as tree graphs.

2) In the numerical calculation of differential cross section, momenta and spins of external particles are fixed to specific values. Thus even if they are identical particles, they are considered to be distinguishable. This means that those graphs, which are equivalent each other under the exchange of identical external particles, should be enumerated as different graphs; they are considered to be topologically different objects. Under this condition, more graphs are generated than the case where the identical particles are not distinguished. The generated source code for numerical calculation is longer than the latter case, but it will be much easier to deal with.

3) Tadpole diagrams are not generated by the current version. Since whether tadpoles is needed or not depends on the prescription of renormalization, it may be required to consider in the future development of automatic system for one-loop calculations.

4) Vacuum-to-vacuum graphs are not considered. We consider only those graphs which have at least two external particles and one vertex.

In this section, we first define some technical terms and discuss some properties of graphs from the graph theoretical point of view. Then the method of graph generation is described.

2.5.1 Notations

Several technical terms have been used in graph theory Ref. [7] but some of them have different meaning between physicists and graph theorists. For these terms we follow physicist's terminology.

In order to help for understanding the terminology, we take a simple graph as shown in figure 2.2, where the points 1, 2, ... 8 are "*nodes*". The "node" is either external particle or vertex.

The connected pair of two nodes, *e.g.* (1,3), (4,5), *etc.*, is called an "*edge*". An edge represents either a propagator between two vertices or a line connecting between an external particle and a vertex. Two nodes, connected by an edge, are called "*adjacent nodes*" of the edge, *e.g.* nodes 1 and 3 are adjacent nodes of edge (1,3). If some distinct edges have a common end node, they are "*adjacent edges*" of the node, *e.g.* edges (1,3), (3,6) and (3,4) are adjacent edges of node 3.

Fig. 2.2 Examples of tree graphs

The "*degree*" of a node is the number of adjacent edges to the node, *e.g.* degree of node 3 is three. The degree of an external particle is one, *e.g.* that of node 2 is one. We also use the word "*the number of legs*" as the same meaning as the degree.

When two nodes are connected through a series of edges, this object is called a "*path*", *e.g.* (1,3,4,5,8) is a path connecting nodes 1 and 8. The "*length of a path*" is the number of edges on the path. The "*distance*" between two nodes is the length of the shortest path joining them. If there is no such a path, distance is said to be infinite.

A graph is "*connected*" if every pair of edges are connected by a path. If a graph is not connected, it is decomposed to several "*connected components*".
The following relation is easily proved for a graph:

Theorem 1 *If a graph has N nodes, I edges, L loops and C connected components, then*

$$N - I + L \;=\; C. \tag{2.171}$$

Proof
Let us first consider a connected graph with no loop ($L = 0, C = 1$). The identity is proved by mathematical induction on the number of nodes N.
When there are two nodes and one edge, the identity holds.
Assuming the identity holds for N node, the number of edge is $I = N - 1$. Then we consider the case of $N + 1$ nodes.
If the $(N + 1)$-th node is an external particle, then it should be connected to one of vertices (not the other external particles). This results in increase of number of edges I by one. The identity holds for this case.
If the $(N + 1)$-th node is a vertex, it should sit on one of existing edge. This vertex divides one edge into two edges. Again the identity holds for this case.

Since these two cases cover all possibilities, the identity holds for $N+1$ nodes case.

Next we prove it by mathematical induction on the number of edges I with fixed number of nodes. Adding an edge to a connected graph, this results in increase of number of loops L by one. Then the identity is also proved in this case.

Since the identity holds for all connected components $C = 1$, the identity for a disconnected graph, composed of several connected components, is proved by summing the identity for each connected component.

Let us consider a graph with E external particles and V vertices, then the number of nodes is given by

$$N = E + V. \tag{2.172}$$

The total number of legs is always equal to twice number of edges. Because each edge connects between two nodes and uses one leg of each connected node. Since an external particle has only one leg, then among the numbers of legs $deg(v)$ of vertex v, external particles E, and edges I the following equation holds:

$$2I = \sum_v deg(v) + E. \tag{2.173}$$

From Eqs. (2.171), (2.172) and (2.173), the following equation is easily proved.

$$E + 2L - 2 = \sum_v (deg(v) - 2). \tag{2.174}$$

We cite the following theorem on a "tree" graph without proof, which is a connected graph with no loop.

Theorem 2 *The following statements are equivalent for a graph G :*

(1) G is a tree graph.

(2) Every two nodes of G are joined by a unique path.

The "*root*" of a tree is a node distinguished from the other nodes. Any external particle can be selected arbitrarily as the root. The graph (a) in figure 2.2 is topologically equivalent to the graph (b), where node 1 is selected as the root. From this theorem, there is an unique path from the root r to any node v. The distance between r and v is determined by the length of the path. This distance is called as the "*depth*" of the node v, which is expressed by $dep(v)$, *e.g.* $dep(\text{node8}) = 4$. The "*father*" of node v is the node which is adjacent to v and at the depth $dep(v) - 1$, *e.g.* father of node 8 is node 5. The root has no father. When a node w is the father of a node v, v is a "*son*" of w. A "*leaf*" is a node with no son, *e.g.* nodes 2.6.7 and 8 are leaves.

A "*blob*" is a one-particle-irreducible (1PI) part including loops. Decomposing a graph into blobs, we can regard it as a tree graph after replacing blobs by vertices. We call such a tree graph "*skelton graph*". In our method the skelton graphs are generated at first, and then the inside structure of each blob is constructed.

2.5.2 Algorithm to generate graphs

The method of Feynman graph generation consists of the following steps:

1) Generation of nodes.

2) Connection of nodes to construct skeleton graphs.

3) Generation of loop structure.

4) Particle assignment.

We describe our method in this order in the following.

Generation of nodes

We consider skeleton graph and ignore the inside structures of blobs.
In the usual field theory, three and four point vertices satisfies the following relation between the number of legs $deg(v)$ for tree vertex v and the order of coupling constant $O(v)$:

$$deg(v) \quad = \quad O(v) + 2. \tag{2.175}$$

Eliminating $deg(v)$ in Eq.(2.174), we obtain

$$E - 2 \quad = \quad \sum_v O(v) - 2L. \tag{2.176}$$

When this graph is regarded as the inside structure of blob vertex w, E and $\sum_v O(v)$ should read as the number of legs $deg(w)$ and the order of coupling constant $O(w)$ of the blob vertex, respectively. Hence we obtain a generalized relation both for the tree and blob vertices w:

$$deg(w) \quad = \quad O(w) - 2L(w) + 2, \tag{2.177}$$

where $L(w)$ is the number of loops defined inside of the vertex w.

Now let us consider skeleton graphs. Since skeleton graph is defined as a tree graph, in which loops are confined in the blob vertices, we obtain basic relations necessary to generate vertices:

$$L \quad = \quad \sum_v L(v), \tag{2.178}$$

$$E - 2 \quad = \quad \sum_v (deg(v) - 2), \tag{2.179}$$

$$O(v) \quad = \quad deg(v) - 2 + 2L(v), \tag{2.180}$$

where L is the total number of loops confined in blobs.

The algorithm to generate vertices for a given number of external particles E and a total order of coupling constants O is the following:

Algorithm 1 *Generation of nodes.*

1. *Determine the number of loops in the final graph by the equation*

$$L = \frac{O - E + 2}{2}. \qquad (2.181)$$

2. *If $L = 0$ then skip to step 3. Otherwise express L as a sum of positive integers, i.e. $L = \sum_i^n l_i, l_i \geq l_{i+1} \geq 1$ where n varies from 1 to L. Since there are many possibilities of this partitioning, the following steps are repeated for all possible partitions. The number l_w is considered as the number of loops $L(w)$ in blob w. The number of blobs is equal to n.*

3. *Determine $deg(v)$ for each vertex by dividing the number $E - 2$ into all vertices according to Eq. (2.179). We first determine the value of $deg(w)$ for each blob ($L(w) \neq 0$), then the number $E - 2 - \sum_{w:blobs}(deg(w) - 2)$ is allotted to tree vertices vs, whose values of $deg(v)$ are limited to either 3 or 4.*

4. *The order of coupling constants of each vertex is determined from $deg(v)$ and $L(v)$ by Eq.(2.180).*

There are many sets of vertices which are generated by this algorithm. The program enumerates all possibilities.

Generation of skeleton graphs

The generated nodes are connected to construct skelton graph by the following algorithm. We take an external particle as the root of the graph.

Algorithm 2 *Connection between nodes.*

1. *Vertices are classified by the numbers of legs and loops.*

2. *Repeat the following steps from 2-1 to 2-3 until all external particles are connected to vertices.*

   2-1. *The vertices in each class are numbered.*

   2-2. *Connect an external particle to a vertex. Since those configurations, where an external particle is connected to different vertices in the same class, are mutually indistinguishable, the vertex with the smallest number is selected not to produce the duplicated graphs.*

   2-3. *The connected vertex is removed from the class and is regarded as a new class which has only this vertex as an element.*

3. *Repeat the following steps from 3-1 to 3-5.*

- *3-1. The remaining vertices in each class are numbered.*
- *3-2. Find a vertex v with just only one free leg, which is not yet connected to another vertex.*
- *3-3. If there is no such a vertex then go to 4.*
- *3-4. Connect v to another vertex. The partner of the connection is selected from each class in the same manner as step 2-2.*
- *3-5. The classes of vertices are updated in the same way as step 2-3.*

4. If the graph is a tree graph, it is accepted as a new graph. Otherwise this configuration is discarded.

Vertices are classified throughout this algorithm, in such a way that topologically equivalent vertices belong to the same class. Since all vertices are isolated at the beginning, classification at step 1 is a simple one.

All external particles are connected to vertices in step 2. Since vertices in a class are not distinguishable, those configurations are not topologically equivalent each other, which are constructed by connecting an external particle with different vertices of a class. So we select only one vertex from each class of vertices. To make the selection systematic, vertices are numbered at step 2-1 and the vertex with the smallest number in its class is selected. There are still a number of ways of selecting a vertex corresponding to the number of classes. They are enumerated recursively.

The connected vertex becomes distinguishable, as it is adjacent to a distinguished external particle. It creates a new class being removed from the old class. Repeating steps from 2-1 to 2-3, some external particles may be connected to the same vertex.

At the beginning of the step 3, let us assume that the obtained configuration C is possible to be a tree graph G by continuing the connecting process. In the graph G, a vertex v with the largest depth from the root is connected to at least $deg(v) - 1$ external particles, since its all sons are external particles. If the father of v is an external particle, the tree graph G has only one vertex v and the configuration C is the same graph as the final form of the graph G. Otherwise, the vertex v in the configuration C has only one free leg. Such a vertex is searched at step 3-2. Let us consider a subgraph composed of this vertex and its adjacent external particles. When such a subgraph is regarded as single node, it can be considered as a kind of an external particle, since it is distinguished from other node and has only one free leg. So the same way of connecting process as step 2 is applied. In this process there appears no equivalent configuration.

This process terminates either when a tree graph is obtained or a configuration is generated without possibility to be a tree graph. They are checked in step 4.

Generation of loop structure

This step constructs the loop structure inside of blob vertex. A blob vertex b has several legs, which are connected to other nodes $n_1, n_2, ..., n_k$ in the preceding subsection. They are distinguished from each other.

Since we consider only one-loop graphs, the loop structure is limited to a circle with several legs on it. What we must do is to put necessary number of tree vertices on the circle and connect their legs to the adjacent nodes of the blob.

A circle is invariant under rotation and reflection. We put tree vertices $v_1, v_2, ...v_m$ along the circle. Each of them is either 3- or 4-point vertices. When all of them are the same type, these symmetries survive. However different types of vertices are mixed, the symmetries are broken. We break these symmetries so as to avoid to generate topologically equivalent graphs. Rotational symmetry of the circle is broken by connecting an adjacent node n_1 and a vertex v_1 on the blob. After this, we put other tree vertices $v_2, ..., v_m$ on the circle in all possible ways.

Next we permutate adjacent nodes $n_1, n_2, ..., n_k$ in all possible ways with fixing n_1, and connect them to $v_1, ..., v_m$ in this order. If v_i is 4-point vertex and n_p, n_q are connected to v_i, we avoid permutations which exchange n_p and n_q. If the circle structure is invariant under reflection symmetry, we avoid to generate permutations which are obtained by exchanging v_i and v_{m-i+2} for all i.

When there are only two vertices on the circle, reflection symmetry cannot be broken. It can be a source of duplicated graph generation in the particle assignment process mentioned later. However, it is simple and straightforward to solve the problem.

Particle assignment

This step is straightforward. Particles are assigned to propagators in the consistent way to the given Feynman rules and external particles. There are several combination of particles assignable to the propagators. In order to make the algorithm effective, we keep a list of candidates of particles for each edge. This kind of list contains only possible combinations, which is checked whether consistent adjacent vertices can be constructed or not. The program searches for an edge with minimum length in its list, and assigns particle to the edge.

Here we obtain the resultant Feynman graphs. All the possibilities is enumerated by back tracking method and by recursive calling of sub-programs.

2.6 Kinematics

As noted in the Introduction (section 1.2.3), kinematics is *not* generated by GRACE but is left to the user. The reason is as follows; due to the lack of generality of the algorithm (see section 2.7.4) used in the current Monte Carlo integration package, the choice of integration variables to get a reliable result is generally highly dependent on the structure of singularities in the amplitude. These are mass-singularity, infrared divergence, t-channel photon exchange and resonance formation which decays into lighter particles. At present, it is quite difficult to prepare the most general kinematics suitable enough for any process, which contains one or some of above singularities. Therefore the user has to write the kinematics so as to change the singular integrand into more smooth one. For the details of the integration algorithm by BASES, we refer to the next section 2.7.

GRACE generates templates of all the subroutines necessary for phase space integration by BASES. These are

1) USERIN initializes BASES.

2) KINIT initializes the kinematics.

3) KINEM calculates four-momenta of final particles from integration variables.

Among these the user is requested to complete USERIN and KINIT by giving some parameters and KINEM by writing kinematics. Others are generated in a complete form (see section 3.3, 3.5 and 4.1) and can be used without any modification unless the user wants to change the default values of parameters included.

BASES is an adaptive Monte Carlo integration package for multi-dimensional variables (see section 2.7 for the detail). The maximally allowed dimension of integration is 50, but only up to 15 variables are allowed for singular variables. The idea for this asymmetric treatment of variables is as follows. In actual problems it would be sufficient if one can make integration up to 6-body primary process, which corresponds to 13 integration variables(assuming the symmetry around the beam axis). After the production of primary particles, such as heavy bosons, one wants to decay these particles. Then one needs other integral variables, which are not singular, to describe these decay processes. Hence 50 variables are grouped into singular and non-singular variables and the user should assign each variable to either of these two categories.

The total dimension of integration, of course, depends on the process considered. It is defined by the FORTRAN variable NDIM in the subroutine USERIN. Declaration of singular and non-singular variables, the upper and lower bounds for each integration variables should be defined in the same subroutine. BASES calls USERIN at the beginning of the integration, and the latter calls KINIT. In this subroutine constant parameters for kinematical variables, such as energy of the system, conditions of experimental cuts and so on, are prepared. The main part of the kinematics should be written in the subroutine KINEM.

Let us consider a process,

$$p_1 + p_2 \to q_1 + \cdots + q_n. \tag{2.182}$$

The cross section is given by (see section 2.1)

$$\sigma = \frac{1}{\text{flux}} \int \prod_i \frac{\mathrm{d}^3 q_i}{2q_{i0}(2\pi)^3} (2\pi)^4 \delta(p_1 + p_2 - q_1 \cdots - q_N)|T|^2. \tag{2.183}$$

We can say that the kinematics is a set of transformations of variables: In the subroutine `KINEM` all the four-momenta and Lorentz invariants should be expressed by the array of integral variables `X`(random numbers) fed by `BASES`. In the generated program the energy of a particle is assigned to the 4-th component of the array `P` which expresses the four-momentum. First three components are spacial momentum. The invariants are assigned to an array `PP`. Random numbers of dimension `NDIM` generated by `BASES` are changed into components of momenta or invariants composed of them for the final state:

$$\texttt{X} \Longrightarrow \texttt{P} \quad \text{or} \quad \texttt{PP}. \tag{2.184}$$

Thus the cross section is rewritten as the integral over `X`

$$\sigma = \int \prod_j \mathrm{d}x_j F(x_1, \cdots, x_{\texttt{NDIM}}), \tag{2.185}$$

and the integrand is constructed from necessary and sufficient number of transformations of the forms

$$\begin{aligned} q_i &= f_i(x_1, \cdots, x_{\texttt{NDIM}}), \qquad (2.186)\\ q_i \cdot q_j &= g_{ij}(x_1, \cdots, x_{\texttt{NDIM}}), \end{aligned}$$

taking into account the constraint by the original δ-function. (Needless to say, the left-hand sides of these equations can be determined in a successive way; they do not need to be explicitly expressed in terms of only x's at the same time.)

The most severe singularity of the amplitude comes out when massless particles appears. Hence photon or gluon are responsible for this in the actual case. In the following we give several typical examples to show how to relate `X` to kinematical variables to smear out singularities by taking e^+e^- reactions.

Infrared divergence

Suppose we have an integral over a real photon(or gluon) energy k,

$$I = \int_{k_{min}}^{k_{max}} \frac{\mathrm{d}k}{k} f(k), \tag{2.187}$$

where $f(k)$ is any function regular at $k = 0$, but $f(0) \neq 0$. This form of integral appears in the phase space integral as

$$\int \frac{\mathrm{d}^3 k}{2k(2\pi)^3} \, |T|^2 \, \cdots \sim \int \frac{\mathrm{d}k}{k} \, (k^2 |T|^2) \, \cdots, \tag{2.188}$$

because $k^2|T|^2$ is finite as $k \to 0$ when a photon(gluon) is emitted. By making the change of variable

$$k = k_{min} \left(\frac{k_{max}}{k_{min}} \right)^x, \tag{2.189}$$

we find that the original integral becomes

$$I = \ln \frac{k_{max}}{k_{min}} \cdot \int_0^1 \mathrm{d}x f(k(x)). \tag{2.190}$$

The singular behavior around $k \sim 0$ is absorbed into the definition of new variable x. Thus this variable can be assigned to one of `X`.

Mass singularity

As a typical example of this kind of singularity, let us consider a real photon emission from e^+e^- colliding beams as shown in figure 2.3. The singularity appears from the propagator of electron(positron) after emitting the photon. This has the form

$$\frac{1}{(p_- - k)^2 - m^2}, \tag{2.191}$$

where p_- is the momentum of e^- and k is that of photon. The electron mass is denoted as m. The denominator can be modified as

$$(p_- - k)^2 - m^2 = -2p_- \cdot k = -2kp \left(\frac{m^2}{p(E+p)} + (1 - \cos\theta) \right). \tag{2.192}$$

Fig. 2.3 Initial radiation diagrams.

Here $p_- = (E, 0, 0, p)$ and $\cos\theta$ is the angle between photon and electron. As $\cos\theta \to 1$ the denominator becomes very small because of $m^2/[p(E+p)]$. In the same way the denominator of the positron propagator can be written as

$$(p_+ - k)^2 - m^2 = -2p_+ \cdot k = -2kp \left(\frac{m^2}{p(E+p)} + (1 + \cos\theta) \right), \tag{2.193}$$

with $p_+ = (E, 0, 0, -p)$. Now we consider the integral of the product of these propagators over photon angle θ.

$$\begin{aligned} I &= \int_{-1}^{1} \mathrm{d}\cos\theta \frac{f(\cos\theta)}{4(p_- \cdot k)(p_+ \cdot k)} \qquad (2.194) \\ &= \frac{1}{4(kp)^2} \int_{-1}^{1} \mathrm{d}z \frac{f(z)}{[m^2/(p(E+p)) + (1-z)][m^2/(p(E+p)) + (1+z)]}. \end{aligned}$$

The integrand contains two very sharp peaks at $\cos\theta = \pm 1$. A change of variable which can absorb these singularities is given by

$$z \equiv \cos\theta = a \cdot \frac{\xi - 1}{\xi + 1}, \tag{2.195}$$

where

$$\begin{aligned} a &= 1 + \frac{m^2}{p(E+p)}, \\ \xi &= \eta^{2x-1}, \qquad (2.196) \\ \eta &= 1 + \frac{2p(E+p)}{m^2}. \end{aligned}$$

The integral I is expressed by new variable x as

$$I = \frac{1}{4(kp)^2} \cdot \frac{1}{a} \ln\eta \int_0^1 \mathrm{d}x \; f(\cos\theta(x)). \tag{2.197}$$

In this way one can deal with both of mass singularities emerging from the initial electron and positron beams at once. Note that a single variable x is enough to manage *two* singularities, thanks to the fact that initial $e^\pm$ beams are coming in the opposite direction. The singular terms appear in the square of amplitude like

$$|T|^2 \sim \frac{m^2}{(p_- \cdot k)^2} \quad \text{and} \quad \frac{1}{p_- \cdot k}. \tag{2.198}$$

(Of course there are similar terms with p_+.) The singularity of the first form is stronger than the second, but the mass squared m^2 in the numerator, suppresses the contribution, and the transformation given above is enough to handle with it.

Throughout this manual we take the process $e^+e^- \to W^+W^-\gamma$ as the example to explain the details of the functions of various parts of the program. In section 3.3 we show the list of the kinematics for this process. We recommend the user to look this list to understand how the infrared and mass singularities are treated in the actual calculation.

t-channel singularity

Consider two peripheral diagrams of two photon process in QED(figure 2.4.). Denoting the squared photon momentum as t (there is of course another t', but things go in the same way), one has the original integral

$$I = \int_{t_{min}}^{t_{max}} \frac{\mathrm{d}t}{t} f(t).$$

It is enough to change the variable according to

$$I = \ln\frac{t_{max}}{t_{min}} \cdot \int_0^1 \mathrm{d}x f(t(x)),$$

when $f(t)$ is explicitly written in terms of

$$1, \quad \frac{m^2}{t}, \quad \frac{1-\cos\theta}{t}, \quad \frac{\sin\theta}{t}, \tag{2.199}$$

where θ is the angle of the scattered particle from the incident one after emitting the virtual photon. All of these terms are not so much singular. When one wants to include Z^0 exchange as well as photon, however, the integration by x gives bad convergence. This is because the peak due to photon exchange is so singular, the Z^0 propagator cannot be well estimated. In this case one has to transform the variable taking both propagators into account. This implies that one has to transform

$$\frac{\text{const.}}{t(t-M_Z^2)}$$

into one integral variable, instead of transforming only $1/t$.

Fig. 2.4 Peripheral two photon diagrams.

Warning of numerical instability caused by t-channel photon

We have to warn the user that in some cases the generated amplitude containing virtual t-channel photon may be numerically not accurate. This happens when the mass of the system created by virtual photon and the other light particle can be very small as shown in figure 2.4. Examples are

$$e^+e^- \rightarrow e^+e^- f\bar{f}, \tag{2.200}$$

$$e^+e^- \rightarrow e^+e^-\gamma, \tag{2.201}$$

with f being a particle lighter than several GeV at $\sqrt{s} \sim 1000$ GeV. In table 2.1 we show test runs for several mass cases of $f = \mu^-$ at $\sqrt{s} = 1000$ GeV. The origin of this instability is the strong cancellation which occurs among amplitudes. To avoid this cancellation we have to use the current conservation at the photon vertex, but in the present version of `GRACE` this kind of protection against cancellation is *not yet made.* We have checked for the case $f = \mu^-$ that we can get correct answer in the quadruple precision as shown in the table. The column indicated by `REDUCE` shows the results of completely independent calculation in double precision with careful treatment of cancellation.

m_f(GeV)	double precision	quadruple precision	`REDUCE`
0.10566	$6.6538 \pm 0.5541 \times 10^5$	$4.2900 \pm 0.0236 \times 10^5$	$4.2661 \pm 0.0180 \times 10^5$
1.0	$2.9527 \pm 0.0350 \times 10^3$	$3.4108 \pm 0.0094 \times 10^3$	$3.3926 \pm 0.0153 \times 10^3$
5.0	$8.7012 \pm 0.0185 \times 10^1$	$9.0786 \pm 0.0227 \times 10^1$	$9.0356 \pm 0.0278 \times 10^1$
10.0	$1.7372 \pm 0.0034 \times 10^1$	$1.7657 \pm 0.0042 \times 10^1$	$1.7632 \pm 0.0054 \times 10^1$

Table 2.1 Cross sections(pb) for the diagrams of Fig.2.4 calculated in different precisions.

Resonance formation

When a particle is produced as a resonance, which decays into other particles, the amplitude contains the propagator

$$\frac{1}{q^2 - M^2 + i\Gamma M} = \frac{q^2 - M^2 - i\Gamma M}{(q^2 - M^2)^2 + \Gamma^2 M^2}, \tag{2.202}$$

where M and Γ are mass and decay width of the particle, respectively and q is its momentum. If this resonance is sharp, one should take q^2 as one of integral variable, and further make the following transformation:

$$I = \int_{q^2_{min}}^{q^2_{max}} \mathrm{d}q^2 \frac{f(q^2)}{(q^2 - M^2)^2 + \Gamma^2 M^2} = \frac{b}{\Gamma M} \int_0^1 \mathrm{d}x\ f(q^2(x)),$$

where

$$\begin{aligned} a &= \tan^{-1}\frac{q^2_{min}-M^2}{\Gamma M}, & (2.203)\\ b &= \tan^{-1}\frac{q^2_{max}-M^2}{\Gamma M}-a, & (2.204)\\ q^2 &= M^2+\Gamma M\tan(a+bx). \end{aligned}$$

In `GRACE`, decay widths of heavy particles are automatically included in *some* of their propagators. The finite width always violates the gauge invariance of amplitude. Nevertheless we modify the propagators according to the rules

1. s-channel propagators contain non-zero finite width.

2. t-channel propagators *do not* contain width.

Here we distinguish s- and t-channel resonance whether the particle is produced and decays directly into lighter particles or not. This different treatment is coming from the observations(which were found empirically rather than theoretically at the moment):

1. Without decay width in s-channel, the cross section diverges at the pole.

2. Finite width in t-channel propagator makes the cross section violently divergent at sufficiently high energies.

The first is an obvious fact but the second catastrophe is unexpectedly large. Thus the width is introduced only in the s-channel propagators. One must keep in mind that the gauge invariance is violated by the inclusion of finite width. This is particularly important to remind when one makes the gauge invariance check of the generated amplitude. The violation in this check is usually in comparable order of $O(\Gamma/\sqrt{s})$.

These are typical examples of how to make singularities harmless by changing the integral variables. We have to make some comments further.

Comments

It is not always possible to assign one integral variable to one singularity. In other words, if the number of singularities exceeds that of independent variables, one has necessarily diagonal singularity. Two photon process including all possible diagrams, $e^+e^- \to e^+e^-\mu^+\mu^-$(or e^+e^-), or radiative Bhabha scattering, $e^+e^- \to e^+e^-\gamma$, are well known examples. If this happens, one possible way to get rid of the difficulty is to divide the whole phase space into two or more regions, and use different set of transformations of independent variables in each sub-phase space. An example of such kinematics will be found in Ref.[8] for the radiative Bhabha scattering. Another may be to discard the part of phase space in which singularities show up and dominate the cross section. This may be not satisfactory, but in massless QCD we have no other way than this.

With keeping the above arguments in mind, one should try to write the most appropriate kinematics for the process to be considered. It is necessary to analyze carefully the structure of singularities and to find the way to avoid diagonal singularities. It might happen that one user succeeded to get good integration for a process, but the other one failed because of an inappropriate kinematics used. One can find the full listing of the kinematics for the radiative process $e^+e^- \to W^+W^-\gamma$ in section 3.3. This may be helpful to learn how to write kinematics.

Inclusion of structure function

In some cases one has to convolute a function with the cross section of sub-process. Typical example is inclusion of structure function of nucleon, radiator function for initial state radiation from e^+e^- beams or luminosity function for equivalent photon approximation. All of these cases can be written symbolically as

$$\sigma(s) = \int \mathrm{d}x F(x,s)\sigma_0(x,s), \tag{2.205}$$

where σ_0 is the cross section of sub-process and $F(x,s)$ represents the function to be convoluted. Since the former is generally given by a multi-dimensional integral over the phase space, $x_1, \cdots, x_{\mathtt{NDIM}}$, we can extend the space to include x and make `NDIM`+1 dimensional integration by `BASES` at once.

2.7 Numerical integration

2.7.1 Integration algorithm

The integration program package BASES uses the importance sampling method. To illustrate this method simply, we consider the following one-dimensional integral;

$$I = \int_0^1 f(x)\mathrm{d}x. \tag{2.206}$$

By modifying this as

$$I = \int_0^1 \frac{f(x)}{p(x)} p(x)\mathrm{d}x \qquad \text{and} \qquad \int_0^1 p(x)\mathrm{d}x = 1, \tag{2.207}$$

the integral can be interpreted as *the expectation value of the function $f(x)/p(x)$ with the probability density function $p(x)$.* The estimate of integral is given by

$$I = E_p[\frac{f(\zeta)}{p(\zeta)}] \quad \simeq \frac{1}{N}\sum_{i=1}^{N} \frac{f(\zeta_i)}{p(\zeta_i)}, \tag{2.208}$$

where ζ is a random number with the probability density function $p(\zeta)$. The variance of the estimate is

$$var.(I) = (\frac{1}{N})^2 \sum_{i=1}^{N} [\frac{f(\zeta_i)}{p(\zeta_i)} - \frac{1}{N}\sum_{j=1}^{N}\frac{f(\zeta_j)}{p(\zeta_j)}]^2 \quad \simeq \frac{1}{N}[\int_0^1 \frac{f(x)^2}{p(x)}\mathrm{d}x - I^2]. \tag{2.209}$$

If the probability density function were given by

$$p(x) = |f(x)| / \int_0^1 |f(y)|\mathrm{d}y, \tag{2.210}$$

the variance could be minimized (zero for a positive definite $f(x)$). Since, however, it is impossible to find such a probability density function, the function $p(x)$ is taken in practice to satisfy $|f(x)|/p(x) \simeq$ constant. In other words, the probability density function $p(x)$ is chosen to be large (*many sample points are thrown*) where $|f(x)|$ is large.

We consider a grid, composed of N_g intervals with variable widths (*small-regions*), over the integration region [0, 1] and sample a small-region with an equal probability $1/N_g$. An equation

$$\frac{1}{N_g} = p(x_i)\Delta x_i \tag{2.211}$$

holds, where Δx_i and $p(x_i)$ are width of the i-th small-region and the probability density function at a point x_i in it, respectively. Therefore the probability density function is given by

$$p(x_i) = \frac{1}{N_g \Delta x_i} \qquad \text{and} \qquad \sum_{i=1}^{N_g} p(x_i)\Delta x_i = 1. \tag{2.212}$$

In BASES, the probability density function is adjusted so as to satisfy the condition

$$\{\frac{f(x)}{p(x)}\}^2 = \{f(x)\Delta x N_g\}^2 \simeq \text{constant.} \tag{2.213}$$

In the multi-dimensional case these small-regions construct a hypercube, *e.g.* a rectangle in the two dimensional case and a rectangular solid in three dimensions. Since the number of small-regions per variable N_g is set to be around 50 in BASES, there are $N_g^{N_{dim}}$ hypercubes, where N_{dim} is the number of dimensions. This is a very large number. Our event generation program SPRING needs a probability information for all hypercubes, according to which a hypercube is sampled. It requires a huge storage space and moreover calculation of the probabilities for all hypercubes needs a lot of computing time. This is the reason why we consider a medium size of regions (*sub-regions*), by which we construct the *hypercube*s for the event generation. Number of sub-regions N_s is determined so that the number of hypercubes $N_s^{N_{dim}}$ should not exceed a given memory size and N_s could divide N_g. In the 10 dimensional case, as an example, there are 2 sub-regions and 50 small-regions for each variable, and $N_{cube} = 2^{10}$ $(= 1024)$ hypercubes in total.

As shown in figure 2.5, execution of the program BASES consists of the grid optimization step and integration step. In the former step, the grids are adjusted as follows;

(1) At the first iteration, all grids have an uniform interval.

(2) The integral is estimated by sampling a set of points for an iteration according to the algorithm mentioned above. During each iteration of estimation, a histogram of $\{f(x)/p(x)\}^2$ is made.

(3) After each iteration, new intervals of the grid are determined so that each interval should have an equal area S/N_g, where S is total area of the histogram.

(4) These procedures (2) and (3) are repeated until the accuracy of the estimate reaches a given value or becomes stable, or number of iterations exceeds a given number.

In the latter step, probabilities for all hypercubes are calculated as well as estimate of integral with the fixed grids, optimized in the former step. This step consists of many iterations of estimating integral and is terminated when the accuracy of the estimate is less than the required one or number of iterations becomes equal to a given number.

In SPRING, a hypercube is sampled according to its probability, prepared by BASES, and a point in the hypercube is sampled and tested. If the grid is enough optimized, even such a simple method can give a good efficiency of event generation.

Before using BASES/SPRING, main program MAINBS and two subprograms USERIN FUNC are to be prepared. In the function program FUNC, a numerical value of the integrand is calculated at the sample point generated by BASES. For numerical calculation of a function, some input parameters, like beam energy, masses of particles etc., are necessary. They are given in `USERIN`, which is called only once at the beginning of the job. Number of integration variables and their lower and upper limits are also given in this subprogram.

Fig. 2.5 Program flow of BASES

2.7.2 Wild variable and BASES50

Recently very high energy e^+e^- colliders, JLC(KEK), NLC(SLAC), CLIC(CERN) and VLEPP(CIS), are studied as future plans. At the energy scale of these machines, the number of final state particles in some elementary processes is too many to apply the original BASES/SPRING directly, because it allows a limited number of dimensions (at maximum 10). Extension of the number of dimensions is easy for the integration (BASES), but is not so easy for the event generation (SPRING).

If we consider the 25 dimensional case, as an example, it is easy to see that a straightforward extension of the dimension number in BASES/SPRING is very difficult. Even though we have two sub-regions for each variable axis, there are $N_{cube} = N_s^{N_{dim}} = 2^{25}$ hypercubes. Calculation of such a huge number N_{cube} of probabilities requires much computing time and their storage is also too large. It seems to be unrealistic.

In order to overcome this situation, a concept of the **wild variables** is introduced. Considering a singular function with many variables, number of those variables, which give the function some singular behaviors, is, in almost all cases, limited. We call this kind of variables the wild variables. If we divide a subspace, spanned by these wild variables, into hypercubes, number of such hypercubes becomes not too large. Thus we apply the BASES algorithm only to this subspace of the wild variables and treat the other variables as additional integration variables. This is a basic idea of the BASES50/SPRING50 algorithm.

In order to show the difference between the algorithms of BASES and BASES50, we consider the following two dimensional integral;

$$I = \int_0^1 f(x,y)\mathrm{d}x\mathrm{d}y = \int_0^1 \frac{f(x,y)}{p(x)q(y)} p(x)q(y)\mathrm{d}x\mathrm{d}y, \tag{2.214}$$

where

$$\int_0^1 p(x)\mathrm{d}x = 1 \quad \text{and} \quad \int_0^1 q(y)\mathrm{d}y = 1. \tag{2.215}$$

By the BASES algorithm, the estimate of this integral is given by

$$I \simeq \sum_{j=1}^{N_{cube}} \frac{1}{N_{trial}} \sum_{i=1}^{N_{trial}} \frac{f(x_i^j, y_i^j)}{p(x_i^j)q(y_i^j)}, \tag{2.216}$$

where (x_i^j, y_i^j) indicate the i-th sample point in the j-th hypercube, $p(x_i^j)$ and $q(y_i^j)$ are the probability densities for the importance sampling and N_{trial} is the number of sampling points per hypercube.

On the other hand, if we take the BASES50 algorithm and consider a variable x as the wild variable, the estimate is given by

$$I \simeq \sum_{k=1}^{N_{cube}} \frac{1}{N_{trial}} \sum_{i=1}^{N_{trial}} \frac{f(x_i^k, y_i)}{p(x_i^k)q(y_i)}, \tag{2.217}$$

where y_i is sampled in the full range of the variable y, while x_i^k is sampled in the k-th subregion.

As shown illustratively in figure 2.6 numbers of hypercubes are N_s^2 and N_s for the former case (a) and the latter case (b), respectively, where N_s is number of sub-regions per variable.

Fig. 2.6 (a) Hypercubes by `BASES` algorithm and (b) those in `BASES50` algorithm, where the variable x is assumed to be wild one.

Although the latter has less number of sampling points for each iteration than the former, it does not affect seriously to the estimate of integral by taking more iterations.

Suppose that N_{dim} and N_{wild} are numbers of integration variables and wild variables ($N_{wild} \leq N_{dim}$), respectively, the number of hypercubes N_{cube} for each case is given by

$$\begin{aligned} N_{cube} &= N_s^{N_{dim}} \quad \texttt{(BASES)} \\ &= N_s^{N_{wild}} \quad \texttt{(BASES50).} \end{aligned}$$

In `BASES50`, the maximum numbers of integration variables and wild ones are equal to 50 and 15, respectively. It is noticed that *as concerns the numerical integration we can obtain the estimate of integral even though the number of those variables, which make the integrand singular, is more than the maximum number of wild variables 15, but for the event generation efficiency may be low.*
Since `BASES50/SPRING50` is the newest version and is recommended to use, **we call** `BASES50 (SPRING50)` **simply as** `BASES (SPRING)` throughout this manual.

2.7.3 `BASES` on a parallel computer

According as increasing the beam energy achieved by accelerators, increases the number of final state particles in the elementary processes to be considered in the experiments. Since it makes generally an expression of the differential cross section very long, the numerical integration requires much computing time. The reason is that the integrand has a huge number of arithmetic operations and is calculated at many sampling points in the phase space.

There are two possibilities to make execution time short. One is to use a **parallel computer**, which consists of many processors with close coupling among them. Distributing the sampling points to N_{node} processors uniformly, the overall execution time can be shorten by a factor N_{node} except for some fraction of an overhead. Since an efficient distribution of sampling points gives an uniform load on each processor, finding a good distribution way is very important for getting a good performance on a parallel computer. Although we can easily imagine several ways, we take, for the time being, the simplest way.

In a parallel version of `BASES`, the distribution is carried out by the unit of hypercube, not of sampling point due to the simplicity of algorithm. If number of hypercubes N_{cube} is a multiple of that of processors N_{node}, we can obtain the best performance in principle. In the case, $N_{cube} < N_{node}$, we lose the power of $N_{node} - N_{cube}$ processors. Therefore it is recommended to control the number of hypercubes N_{cube}, which depends on numbers of the wild variables N_{wild} and of sampling points N_{call}, which are given as input parameters of the integration (see section 3.5.3). The number N_{wild} is normally fixed for an elementary process but the number N_{call} can be arbitrarily chosen.
The program flow of the parallel version is conceptually depicted in figure 2.7(c), while figure 2.7(a) shows that for the scalar version (single `CPU` version). The numerical integration on a parallel computer proceeds as follows:

1) At the beginning of integration, the subprogram `USERIN` is called, where several parameters for integration and integrand are set.

2) At the beginning of the grid optimizing step the grids are set to be uniform, while in the integration step they are fixed as determined by the grid optimization step.

3) For calculating estimate of the integral, the hypercubes are distributed to each processor. If the number of hypercubes N_{cube} is not a multiple of that of processors N_{node}, some processors have one more hypercube than the others. Since these differences are negligible small as long as N_{cube} is much larger than N_{node}, it does not seriously affect to the performance.

4) In each processor, the numerical integration is carried out for those hypercubes distributed by the step (3). During this procedure a histogram of the function $\{f(x)/p(x)\}^2$ is made in the grid optimization step.

5) After the integration in each processor is terminated, the estimate of each hypercube is transferred from each processor to the central processor, where the current and also cumulative results of the integral are calculated.

6) If the cumulative results of integration fulfill the convergency condition, either the grid optimization step becomes the integration step and go to step (4) or the integration step is terminated and go to the end.

7) In the grid optimization step, a histogram information of the function $\{f(x)/p(x)\}^2$ in each processor is also transferred to the central processor, where a new grid is determined by using the histogram content, and then go to step (4).

Fig. 2.7 The conceptual program flows of `BASES` for (a) the scalar version, (b) the vector version and (c) the parallel version.

Fitting of the execution time $T_{N_{node}}$ with N_{node} processors to an empirical formula

$$T_{N_{node}} = O + \frac{Q}{N_{node}} \tag{2.218}$$

is quite good. The quantity O may give the overhead of integration, which consists of calculating the cumulative results, data transfer from each processor to the central one and adjusting the widths of grids. From this formula we can see that if the quantity O is negligible small comparing to the quantity Q/N_{node} the more processors make the execution time the shorter. But if O is comparable to Q/N_{node} increasing number of processors is not effective.

Another possibility to make the execution time short is to use a vector computer. The conceptual program flow of the vector version is shown in figure 2.7(b), where hypercubes are grouped into N_v groups similar to the parallel version. The amplitude calculation for each group is performed by a single `CALL` of the subroutine `VBFNCT` with the vectorized code, while it is performed on different node in the parallel version. More detailed description of the vector version is appeared in chapter 5.

2.7.4 A weak point in `BASES` algorithm

There is a crucial restriction on use of `BASES`. As an example, consider the following two-dimensional integral with a parameter of ϵ:

$$I = \int_0^1 \mathrm{d}x \int_0^1 \mathrm{d}y \frac{2y\epsilon}{(x+y-1)^2+\epsilon^2} \quad \to \ \pi \ (\epsilon \to 0). \tag{2.219}$$

This integrand has a singularity along the line $x+y-1=0$ as shown in figure 2.8 (a).

Fig. 2.8 (a) Singularity on the $x-y$ plane, (b) Singularity on the $X-Y$ plane.

It runs exactly along a diagonal line of the phase space. During the grid optimization step, histogram of $\{f(x)/p(x)\}^2$ is made for each integration variable to determine new intervals of grid for the next iteration. Since, however, any singular behavior may not be seen in the histogram for this case, the new grid is not to be different appreciably from the old one. This means that grid cannot be suited for the behavior of this function. As a result the integration algorithm is identical to the crude Monte Carlo method. If the width of singularity is not narrow, there may be some probability to hit the singularity and hence the estimate of integral may be reliable. Since, however, those singularity with a very narrow width can be rarely hit , an unexpectedly small value of the estimate may be obtained.

We can, of course, obtain a reasonable result when we throw a huge number of sampling points per iteration and use a lot of computing time. Since the integration requires so much computing time even for such a simple function, `BASES` is practically useless for a real process with this kind of singularity.

Changing the integration variables from x, y to the new variables $X(= x - y)$, $Y(= x + y)$ as shown in figure 2.8 (b), where the singularity runs parallel to the X axis, `BASES` can give a good answer for each value of ϵ.

Theses two examples show that the choice of an appropriate set of integration variables is very important to obtain a reliable answer. With an unsuitable combination of variables, the estimate results in a smaller value than the good answer and its error is unfortunately not significantly large. But *the accuracy for each iteration fluctuates, iteration by iteration, and, in some case, it jumps up suddenly to a large value compared to the other iterations.* This is the only indication for taking an unsuitable variable set. *Be careful!*

2.8 Event generation

There are many ideas to generate random numbers with an arbitrary distribution; direct method, rejection method, composition method, composition-rejection method and so on. Since, however, there is no method for general purpose, we have to select or find an appropriate method, case by case. The program package `BASES/SPRING` makes it possible to generate random numbers with an arbitrary multi-dimensional distribution (50 at maximum) efficiently without such a consideration.

The algorithm of `SPRING` is quite simple. By the integration algorithm of `BASES`, the widths of the grid are adjusted so that each interval of the grid contributes equally to estimate of the integral. In the multi-dimensional case, this is approximately true, but not exact. As shown in figure 2.9, the program flow of the event generation by `SPRING` is as follows:

1) Integrate the differential cross section over the phase space by `BASES`. During the integration the maximum value of function $f(x)/p(x)$ as well as the probability for each hypercube is calculated.

2) Sample a hypercube (say i-th hypercube) with its probability.

Fig. 2.9 The program flow of `SPRING`

3) For each wild variable, sample a small-region in the i-th hypercube and sample a point in the small-region. For each non-wild variable, sample a small-region from the full range of the variable and sample a point in the small-region. For this sampling one random number is enough for one variable.

4) If the sampled point ζ satisfies the condition

$$\frac{f(\zeta)}{p(\zeta)}/Max.(\frac{f(x_i)}{p(x_i)}) < \ \eta \ (= \text{ a uniform random number}),$$

then this point is accepted as a generating point, subroutine SPEVNT is called and go to 2). In SPEVNT, users are to save four vectors of the accepted event on a file.

5) If not, forget this point and go to 2).

Even by such a simple algorithm events with the distribution of the differential cross section are easily generated, as long as the differential cross section can be integrated by BASES. If the grid is not enough optimized, the generation loop (from steps 2) to 4)) may come into an infinite loop. To avoid getting into this infinite loop, the maximum number of trials for generating an event is to be given in a main program as a parameter. It should be noted that *the subprograms* USERIN *and* FUNC *are to be identical to those for* BASES except for the case where the integrand is a many-valued function of the integration variables.

Chapter 3

GRACE system

As mentioned in section 1.2.2, **GRACE** system consists of the following four subsystems:

- Graph generation subsystem
- Source generation subsystem
- Numerical integration subsystem
- Event generation subsystem.

In this chapter the specifications of these subsystems are described. Before coming into the details, it may be useful to summarize them briefly here.

Graph generation system

Input :

1) **Definition of physical process**
   Specification method of the physical process is described in subsection 3.1.1.
2) **The model definition file**
   Since specification of model is rather complicated, we postpone its description to chapter 6. We provide a default standard model following Ref.[1], [2] and we recommend to use this model for the first use of this system.

Output :

1) **Graph information file** `OUTDS`
2) **Drawn figures**
   Generated graphs are drawn on a graphic device by using the file `OUTDS`. They are described in section 3.1.2.

Source generation subsystem

Input :

1) **The model definition file**
2) **Graph information file** `OUTDS`
   which is generated by the graph generation subsystem.

Output :

1) **Generated FORTRAN source code**
   There are three kinds of FORTRAN source codes generated by `GRACE`. The first is a set of subprograms for amplitude calculation, whose description is briefly given in section 3.2. These subprograms use the `CHANEL` routines through the interface subprograms. Specification of the interface subroutines and `CHANEL` routines will be described in sections 7.2 and 7.3 respectively.
   The second is a set of subprograms for the integration program `BASES`, which will be explained in section 3.5. The third is a set of subprograms for the event generation program `SPRING` and is described in section 3.6. In section 3.3 the program specifications of kinematics routines are given.
2) **Output of the testing program**
   The format of output of the generated test program is given in section 3.4.

Numerical integration subsystem

Input :

1) **Subroutines for the kinematics**
   We leave writing the kinematics part to users, since it is difficult to generate this part automatically as mentioned in section 3.4. The description of related subprograms is given in section 3.3.
2) **Subroutines for the amplitude calculation**
   These subprograms are generated by the source generation subsystem described in section 3.2.
3) **Subprograms for the integration by** `BASES`
   Before integration by `BASES`, users should prepare subprograms `USERIN` and `FUNC`. Specifications of these subprograms are described in section 3.5, as well as the input parameters for the integration.

Output :

1) **Print out**
   The format of output of `BASES` is given in section 3.5. There may be statistical error in the Monte Carlo integration and systematic error in user's kinematic subroutines. So it is very important to see whether the result is reliable or not.

2) **Probability information file**
As the result of integration, the probability information, contents of histograms *etc.* are saved in this file, which is used for event generation.

Event generation subsystem

Input :

1) **Subroutine for the kinematics**
2) **Subroutines for the amplitude calculation**
3) **Subprograms for the program package** `SPRING`
Subprograms `SPINIT, SPEVNT` and `SPTERM` are required to prepare. In section 3.6 their specifications are given besides the input parameters.
4) **Probability information file**
which is generated by `BASES`.

Output :

1) **Print out**
The print out format is given in subsection 3.6.4. This is very useful to see whether the generated events reproduce really the distribution of differential cross section.
2) **Output file for the generated events**
Generated events are passed to detector simulator or simulator of particle decay. Section 3.6 describes how to deal generated events for this purpose.

The generated FORTRAN code uses default values of mass parameters, coupling constants and other parameters, whose values are set in the subprogram `SETMAS`. If one wants, one can change these values by modifying this subroutine.

Although many physical processes have been calculated for testing the `GRACE` system, it is still possible that a new error may occur in a new reaction. It is important to check the result in a systematic way. Possible origin of error will be

(1) Unsuited kinematical variables to the integrand,

(2) Bugs in the kinematics,

(3) Large numerical cancellation,

(4) Bugs in the `GRACE`.

Numerical cancellation is the most difficult problem to control. Even if the program is logically correct, it is possible to produce completely wrong result. Some of numerical cancellation can be avoided by improving kinematics, but others require modification of generated code.

Anyway one has to check the result intensively. Usual checking method is as follows:

(1) Check gauge invariance of the result,

(2) Check Lorentz frame invariance of the result,

(3) Check numerical stability of the result,

(4) Changing the number of sampling points in the numerical integration,

(5) Comparison with other results.

Before the numerical integration, one should confirm that the generated FORTRAN source code is correct one. GRACE system generates a test program, which provides a gauge invariance test by comparing the resultant values on a phase space point for different values of gauge parameters. One can check some kind of numerical cancellation or inconsistency in the generated code. This is the easiest way of checking. However, since this program checks only at one point, one may miss errors in the different region of the phase space.

Since the amplitude is calculated by a numerical way in a special Lorentz frame, one can test the program by changing reference frame. This method also checks numerical cancellation partially, as the four components of momenta are changed.

Direct checking method of numerical cancellation is to change precision of the calculation. If your compiler has an option to change precision of floating point number, it will easy and powerful method.

The correctness of the kinematics subroutines and statistical reliability will be checked by careful reading of output of BASES and changing parameters of BASES. If kinematics subroutines fails to catch steep peeks of the differential cross section, the final value may be completely wrong.

3.1 Graph generation

3.1.1 Definition of the physical process

In order to define a physical process we give the order of coupling constants and names of external particles as the input.

We show an example, which specifies the process $e^+e^- \to W^+W^-\gamma$, in figure 3.1.

```
* 5120      E+ E-  => W+  W-  A       TREE
WORDER       3
INITIAL EL  1
INITIAL ELB 1
FINAL   WB  1
FINAL   WBB 1
FINAL   AB  1
END
```

Fig. 3.1 Input file for defining physical process

The format of input is as follows:

1) **Comment line**
   The first line is a comment line, but it should *never* be omitted. It is copied to output files to indicate the process.

2) **The order of coupling constants**
   The second line in the example indicates the order of coupling constants.

   ```
   WORDER      3
   ```

   implies that the order of electroweak interaction (order of perturbation) is 3. When one want to restrict the process to pure QED,

   ```
   EORDER
   ```

   should be assigned. It is noted that `WORDER` and `EORDER` are not allowed to set at the same time. For QCD one should give the order of QCD coupling by

   ```
   CORDER.
   ```

   Combination of `WORDER` and `CORDER` or that of `EORDER` and `CORDER` are allowed. In that case the order of each interaction should be defined in different line.

3) **External particles**
   To define the external particles, in the first column one has to give whether the current particle is in the `INITIAL` or `FINAL` state. Then name of this particle follows. If it is anti-particle, `B` should be added to the end of the name. For the W-boson, `WB` defines W^+, so that W^- is written as `WBB`. In the last column the number of identical particles is given by an integer.

Table 3.1 shows a list of particle names defined in the model definition file, whose format is described in chapter 6.

In `UNIX` system, many files named like "`dnnnn`" are given under the directory `$GRACEDIR/data/` as examples of the input file, whose list is in the file "`Index`" (see also section 4.1). The contents of file "`Index`" is given in Table 3.2, where the last three numbers of each line are the orders of perturbation, `WORDER`, `EORDER` and `CORDER`. If there is the target process in this list, the first number `dnnnn` indicate the file name which contains the input parameters for that process like figure 3.1. For example, if one wants to calculate $e^+e^- \to W^+W^-\gamma$, one can use the file `d5120`. When one cannot find the process to be studied, it would be easy to make input file by copying a similar process's file.
The file `particle.table` under the same directory contains all the information on the model used in the graph generation and source generation subsystems.

name of particle	
WB	W^+
ZB	Z^0
AB	γ
XB	χ^+
X3	χ_3
PH	ϕ (Higgs boson)
NE	ν_e
NM	ν_μ
NT	ν_τ
EL	e^-
MU	μ^-
TA	τ^-
UQ	u-quark
CQ	c-quark
TQ	top-quark
DQ	d-quark
SQ	s-quark
BQ	b-quark
CP	c^+ (ghost for W)
CM	c^- (ghost for W)
CZ	c^Z (ghost for Z)
CA	c^A (ghost for photon)
GL	gluon
CG	c^G (ghost for gluon)

Table 3.1 Names of particles in the default model definition file

d3010	A => E+ E-	TREE	0	1	0
d3020	A => U UB	TREE	1	0	0
d3030	W- => E- NUB	TREE	1	0	0
d3040	W+ => U DB	TREE	1	0	0
d3050	Z => E+ E-	TREE	1	0	0
d3060	Z => W+ W-	TREE	1	0	0
d3070	Z => Z Z	TREE	1	0	0
d3080	G => U UB	TREE	0	0	1
d4010	E- E- => E- E-	TREE	2	0	0
d4020	E+ E- => E+ E-	TREE	2	0	0
d4030	E+ E- => MU+ MU-	TREE	2	0	0
d4040	E+ E- => U UB	TREE	2	0	0
d4050	E+ E- => A A	TREE	2	0	0
d4060	E+ E- => W+ W-	TREE	2	0	0
d4070	E- NEB => D UB	TREE	2	0	0
d4080	U DB => U DB	TREE	0	0	2
d4110	W+ W- => W+ W-	TREE	2	0	0

Table 3.2 The list of processes in the file `Index`

continued to the next page

d4090	U	UB	=>	G	G				TREE	0	0	2
d4100	U	UB	=>	A	Z				TREE	2	0	0
d4140	A	A	=>	Z	Z				TREE	2	0	0
d4150	G	G	=>	U	UB				TREE	0	0	2
d4120	W+	W-	=>	Z	Z				TREE	2	0	0
d4130	Z	Z	=>	Z	Z				TREE	2	0	0
d4160	G	G	=>	G	G				TREE	0	0	2
d4170	E+	E-	=>	NE	NEB				TREE	2	0	0
d4180	A	A	=>	W+	W-				TREE	2	0	0
d5010	E+	E-	=>	E+	E-	A			TREE	3	0	0
d5020	E+	E-	=>	E-	NEB	W+			TREE	3	0	0
d5030	E+	E-	=>	MU+	MU-	A			TREE	3	0	0
d5040	U	UB	=>	C	CB	G			TREE	0	0	3
d5050	W+	W-	=>	W+	W-	A			TREE	3	0	0
d5060	Z	Z	=>	Z	Z	A			TREE	3	0	0
d5070	U	A	=>	U	G	G			TREE	0	1	2
d5080	G	A	=>	U	UB	G			TREE	0	1	2
d5090	A	A	=>	U	UB	G			TREE	0	2	1
d5100	U	G	=>	U	G	G			TREE	0	0	3
d5110	E+	E-	=>	E+	E-	Z			TREE	3	0	0
d5120	E+	E-	=>	W+	W-	A			TREE	3	0	0
d5130	E+	E-	=>	NE	NEB	A			TREE	3	0	0
d5140	E+	E-	=>	W+	W-	Z			TREE	3	0	0
d5150	G	G	=>	U	UB	G			TREE	0	0	3
d5160	E+	E-	=>	Z	Z	A			TREE	3	0	0
d5170	E+	E-	=>	NE	NEB	PH			TREE	3	0	0
d5180	NE	MU	=>	E-	NM	A			TREE	3	0	0
d6010	E+	E-	=>	E+	E-	A	A		TREE	4	0	0
d6020	E+	E-	=>	E+	E-	MU+	MU-		TREE	4	0	0
d6030	E+	E-	=>	MU+	MU-	MU+	MU-		TREE	4	0	0
d6040	U	UB	=>	C	CB	G	G		TREE	0	0	4
d6050	W+	W-	=>	W+	W-	W+	W-		TREE	4	0	0
d6060	Z	Z	=>	Z	Z	Z	Z		TREE	4	0	0
d6070	Z	Z	=>	W+	W-	Z	Z		TREE	4	0	0
d6080	E+	E-	=>	U	UB	G	G		TREE	0	2	2
d6090	E+	E-	=>	U	UB	U	UB		TREE	0	2	2
d6100	E+	E-	=>	U	UB	D	DB		TREE	0	2	2
d6110	A	A	=>	D	DB	G	G		TREE	0	2	2
d6120	E+	E-	=>	W+	W-	Z	Z		TREE	4	0	0
d6130	E+	E-	=>	MU+	MU-	A	A		TREE	4	0	0
d6140	A	A	=>	E+	E-	E+	E-		TREE	4	0	0
d6150	E+	E-	=>	W+	W-	NE	NEB		TREE	4	0	0
d6160	E+	E-	=>	W+	W-	E+	E-		TREE	4	0	0
d6170	E+	E-	=>	PH	PH	NE	NEB		TREE	4	0	0
d6180	E+	E-	=>	E+	E-	MU+	MU-		TREE	0	2	0
d7010	E+	E-	=>	D	DB	G	G	G	TREE	0	2	3

Table 3.2 The list of processes in the file `Index`

3.1.2 Drawn Feynman graph

In the graph generation, a file `OUTDS` is created under the current directory, where the graph information is stored. By typing command "`draw`", Feynman graphs are drawn on the screen when the X-Window system is supported. The drawing method is very primitive. There are two kinds of vertecies, those on a fermion line and the others. Vertecies of first kind are placed on the fermion lines, where the positions of fermion lines and vertecies are arranged so that fermion lines could not cross each other. The other vertices are placed at fixed positions in accordance with the number of vertices. The convention of drawing graph is as follows:

1) External particle lines carry the labels of particle name, such as `EL I` or `WB F`. Here `I` and `F` mean initial and final particle, respectively.

2) The arrow attached to the internal line does not indicate fermion number but the direction of the flow of the quantum number $(charge) - 2 * (baryon\ number)$.

The quality of drawn figures are not so good. It will be improved in future after a detailed analysis of graph structure. Figure 3.2 shows drawn graphs for the process $e^+e^- \to W^+W^-\gamma$.

Fig. 3.2 An example of drawn Feynman graphs

continued to the next page

Fig. 3.2 An example of drawn Feynman graphs

continued to the next page

Fig. 3.2 An example of drawn Feynman graphs

3.2 Generated source code

There are three kinds of program components. The first is for the amplitude calculation, the second is necessary for the integration by `BASES` and the third is for the event generation by `SPRING`. The interrelation among the subprograms generated by `GRACE` is depicted in figure 3.3, where those subprograms in the **`white box`** are automatically generated by `GRACE`, while those in the **`shaded box`** are already contained in other program packages `BASES/SPRING`, interface program library to `CHANEL`, and program package `CHANEL`. The program specifications of the libraries `BASES/SPRING`, the interface to `CHANEL` and program package `CHANEL` are described in sections 3.5, 7.2 and 7.3, respectively.

Fig. 3.3 Relation among the generated subprograms

In the following these three kinds of program components are summarized. The most of components used in `BASES` are required also in `SPRING`, so that they appear in the both items.

1) a set of program components for amplitude calculation

`SETMAS`	*(subroutine)*	defines masses and decay widths of particles.
`AMPARM`	*(subroutine)*	defines coupling constants and others.
`AMPTBL`	*(subroutine)*	calls `AMnnnn` to calculate amplitudes.
`AMPSUM`	*(subroutine)*	sums matrix elements over the helicity states. A matrix element is the square of the sum of amplitudes.
`AMnnnn`	*(subroutine)*	calculates amplitude of the `nnnn`-th graph, where the number `nnnn` of the routine name is equal to the graph number.
`AMPORD`	*(subroutine)*	arranges amplitudes.
`incl1.f`	*(include file)*	defines the common variables for masses, amplitude tables *etc.*
`incl2.f`	*(include file)*	defines the work space for `AMPTBL`.
`TEST`	*(main)*	works as the main program for testing gauge invariance.

2) a set of program components for the integration by `BASES`

`MAINBS`	*(main)*	is the main program for the integration.
`USERIN`	*(subroutine)*	initializes `BASES` and user's parameters.
`KINIT`	*(subroutine)*	initializes kinematics.
`FUNC`	*(function)*	calculates the numerical values of differential cross section.
`KINEM`	*(subroutine)*	derives particle four momenta from the integration variables.
`USROUT`	*(subroutine)*	prints the amplitude summary table.
`inclh.f`	*(include file)*	defines the size of the histogram buffer.

3) a set of program components for the event generation by `SPRING`

`MAINSP`	*(main)*	is the main program for the event generation.
`USERIN`	*(subroutine)*	initializes `BASES` and user's parameters.
`KINIT`	*(subroutine)*	initialize kinematics.
`FUNC`	*(function)*	calculates the numerical values of differential cross section.
`KINEM`	*(subroutine)*	derives particle four momenta from the integration variables.
`inclh.f`	*(include file)*	defines the size of the histogram buffer.
`SPINIT`	*(subroutine)*	initializes routine for user's purpose.
`SPEVNT`	*(subroutine)*	saves four vectors on a file.
`SPTERM`	*(subroutine)*	is called at the termination for user's purpose.

Although the program components `inclh.f`, `USERIN`, `KINIT`, `FUNC` and `KINEM` are created automatically by `GRACE`, they are still *imcomplete.* Especially the kinematics routines `KINIT` and `KINEM` are to be filled up by the user according to their specifications described in section 3.3. Example of these routines for the process $e^+e^- \to W^+W^-\gamma$ are also given there.

The subprograms `USERIN` and `FUNC` are used both in the numerical integration by `BASES` and the event generation by `SPRING`. In the user initialization routine `USERIN`, subroutines `KINIT` is called for initializing kinematics part. The routines `SETMAS` and `AMPARM` are called for the initializtion of amplitude calculation.
The function subprogram `FUNC` is used for calculating the numerical value of differential cross section, where subroutine `KINEM` is called for calculating four vectors of external momenta and subroutines `AMPTBL` and `AMPSUM` are called for the amplitude calculation. Since specifications for `USERIN` and `FUNC` are described in sections 3.5.3 and 3.5.4 respectively, in this section we mention the amplitude calculation part briefly.

3.2.1 Initialization of amplitude calculation

Parameters for the amplitude calculation are set in subprograms `SETMAS` and `AMPARM`, and are passed to the relevant subroutines through the several commons, which are given in the include file `incl1.f`.

Subroutine `SETMAS`

The structure of subprogram `SETMAS` is shown in the source list 3.1. In `SETMAS` the following fundamental parameters are defined.

1) **Masses** and **widths** are defined.

2) **Gauge parameter**
The information about the gauge parameters is summarized in the include file `incl1.f`.

Calculation either in covariant gauge(R_ξ-gauge) with an arbitrary gauge parameter or in unitary gauge is possible in `GRACE` system. The distinction between them is given by integer variables in the common `/SMGAUS/`.

```
COMMON /SMGAUS/IGAU00, IGAUAB, IGAUWB, IGAUZB, IGAUGL
```

where `IGAUAB`, `IGAUWB`, `IGAUZB` and `IGAUGL` are the gauge selection flags for photon, $W^\pm$, Z^0 and gluon, respectively. Unitary gauge Eq.(2.101) is selected by setting flag `IGAU`*xx* to `0` for *xx* boson. This is effectively equivalent to the case where the gauge parameter of *xx* boson is set equal to infinity.

For the covariant gauge, four different values of gauge parameters can be set by using an array `AGAUGE(`*i*`)` (*i* runs from 1 to 4).

In the generated FORTRAN code, unitary gauge is taken as the default gauge.

```
COMMON /SMGAUG/AGAUGE(0:4)
REAL*8 AGAUGE
```

`AGAUGE(IGAUAB)`, `AGAUGE(IGAUWB)`, `AGAUGE(IGAUZB)` and `AGAUGE(IGAUGL)` represent the values of gauge parameters α_A, α_W, α_Z and α_G, respectively (see Eq.(2.50)). To give different values of gauge parameters for each boson, the flags `IGAUAB`, `IGAUWB`, `IGAUZB` and `IGAUGL` are to be set equal to 1, 2, 3, and 4, respectively, for example. Of course, the values should be set for the variables `AGAUGE(IGAU`xx`)`s here.

3) **Spin summation**
The components of spin and polarization vector are controlled by

Fermion	: 0 (helicity $= -1$), 1 (helicity $= +1$)
Vector boson	: 0, 1 (transverse), 2 (longitudinal).

For each external particle `I` of non-zero spin, the spin summation is taken from `JHS(I)` to `JHE(I)` as follows;

```
      ANS = 0.0
      DO 100 J = JHS(I), JHE(I)
        ANS = ANS + table_of_amplitude(J)
  100 CONTINUE
```

where

```
      JHS(I) = 0
      JHE(I) = LEPEXA - 1
```

and

```
      LEPEXA = 2
```

for the external photon as an example. In the generated code, the spin summation is originally arranged to give unpolarized cross section. The spin freedoms of external particles are given in the include file `incl1.f` (see Source list 3.3) as follows;

`LEPEXA = 2`	spin freedom of external photon
`LEPEXW = 3`	spin freedom of external $W^{\pm}$ boson
`LEPEXZ = 3`	spin freedom of external Z^0 boson
`LEPEXG = 2`	spin freedom of external gluon
`LEXTRN = 2`	spin freedom of external fermion

The variable `ASPIN` is the normalization factor of spin average for initial bosons and fermions.

4) **Selection of diagrams**
If one sets the i-th element of the array `JSELG( )` to "zero", then one can omit the corresponding i-th graph and skip the calculation of this amplitude. Each element of the array correspond to the graph number which can be read off from the drawn picture of graphs.

```
      SUBROUTINE SETMAS
      IMPLICIT REAL*8(A-H,O-Z)

      INCLUDE 'incl1.f'
      COMMON /AMSPIN/JHS(NEXTRN), JHE(NEXTRN), ASPIN
*----------------------------------------------------------------------
* Graph selection
      DO 10 NG = 1, NGRAPH
        JSELG(NG) = 1
   10 CONTINUE
*----------------------------------------------------------------------
* Mass
      AMWB =  80.0D0
      AMZB =  91.1D0
      AMAB =   0.0D0
      AMEL =   0.511D-3
      AMMU = 105.658387D-3
      . . . . . .                        masss of particle.
* Width
      AGWB = 0.0D0
      AGZB = 0.0D0
      AGAB = 0.0D0
      AGEL = 0.0D0
      AGMU = 0.0D0
      . . . . . .                        decay width of particle.

* Gauge parameters (default is unitary gauge)
      IGAUAB = 0
      IGAUWB = 0
      IGAUZB = 0
      IGAUGL = 0
      AGAUGE(IGAU00) = 1.0D0
      AGAUGE(IGAUAB) = 1.0D0
      AGAUGE(IGAUWB) = 1.0D0
      AGAUGE(IGAUZB) = 1.0D0
      AGAUGE(IGAUGL) = 1.0D0
*
* Spin average                           Control of spin summation.
      ASPIN = 1.0D0
*                                         1:  EL   INITIAL   LPRTCL
      JHS(   1) = 0
      JHE(   1) = LEXTRN - 1
      ASPIN = ASPIN/DBLE(JHE(   1)-JHS(   1)+1)
*                                         2:  EL   INITIAL   LANTIP
      JHS(   2) = 0
      JHE(   2) = LEXTRN - 1
      ASPIN = ASPIN/DBLE(JHE(   2)-JHS(   2)+1)
```

Source list 3.1 subprogram SETMAS

continued to the next page

```
*                                           3:  WB   FINAL     LPRTCL
      JHS(   3) = 0
      JHE(   3) = LEPEXW - 1
*                                           4:  WB   FINAL     LANTIP
      JHS(   4) = 0
      JHE(   4) = LEPEXW - 1
*                                           5:  AB   FINAL     LPRTCL
      JHS(   5) = 0
      JHE(   5) = LEPEXA - 1
*
      RETURN
      END
```

Source list 3.1 subprogram SETMAS

Subroutine AMPARM

In the source list 3.2 the structure of subprogram AMPARM is given, which prepares the following items:

1) **Version number**
   The version number of GRACE system is compared with that of the interface package to CHANEL in SMINIT. If they are not consistent, job is terminated for the sake of safty.

```
      SUBROUTINE AMPARM
      IMPLICIT REAL*8(A-H,O-Z)

      INCLUDE 'incl1.f'
      COMMON /AMCNST/ PI, PI2, RAD, GEVPB, ALPHA
*-------------------------------------------------------------------
      CALL SMINIT(   1,   0)
* Constants
*-------------------------------------------------------------------
      PI    = ACOS(- 1.0D0 )
      PI2   = PI * PI
      RAD   = PI / 180.0D0
      GEVPB = 0.3893857D9
      ALPHA = 7.2973503D-3
      AMWB2 = AMWB*AMWB
      AMZB2 = AMZB*AMZB
      . . . . . . . . .              constant parameters for vertex.
```

Source list 3.2 subprogram AMPARM

continued to the next page

```
*-----------------------------------------------------------------
* VVV
      CZWW    =  CE*GW
      CAWW    =  CE
      . . . . . . . . .                 coupling constants
* QCD coupling constant should be calculated in 'KINIT'.
      CQCD   = 1.0D0
      CQCDSQ = 1.0D0
      CQQG(1) = -1.0D0
      CQQG(2) = -1.0D0
* Color facotr
      DO 100 I = 1, NGRAPH
        IGRAPH(I) = 0
        DO 100 J = 1, NGRAPH
          CF(J, I) = 1.0D0
  100 CONTINUE
*
      RETURN
      END
```

Source list 3.2 subprogram `AMPARM`

2) **Constants**
Numerical constants π, π^2, $\pi/180$, GeV/pb and $\alpha = e^2/4\pi$ for the amplitude calculation are set and some of them are passed through the common `/AMCNST/` for later use.

3) **Coupling constants**
Coupling constants for various vertices are calculated.

4) **Color facotrs**
Color factors (the array `CF`(i,j)) for each combination of two graphs are calculated.

Include file `incl1.f`

This file is prepared for passing the parameters for the amplitude calculation set in the subroutines `SETMAS` and `AMPARM` to the relevant subroutines through the several commons. In the source list 3.3 the structure of `incl1.f` for the process $e^+e^- \to W^+W^-\gamma$ is shown.

1) **Parameter statements**
The parameters which define the sizes of arrays are given by the parameter statement. `LEPEXA`, `LEPEXW`, `LEPEXZ` and `LEPEXG` are the spin freedoms of external photon, W-boson, Z-boson and gluon, respectively. `LEPINA`, `LEPINW`, `LEPINZ` and `LEPING` are those for internal lines. `LEXTRN` and `LINTRN` are the spin freedoms for fermions of external and internal lines, respectively.

The parameters LOUTGO, LINCOM, LANTIP and LPRTCL are just the input constants for the program package CHANEL.

2) **Table of amplitude**
The calculated amplitudes for all graphs are stored in an array AG( ). An array APROP( ) is used to keep the numerical value of the denominators of propagators.

The arrays AV( ), LT( ) and INDEXG( ) in the common /SMATBL/ are for temporary use.

3) **Masses and widths**
The variables in the commons /AMMASS/ and /AMGMMA/ are masses and widths of particles, respectively, which are defined in SETMAS.

4) **Coupling constants**
The coupling constant for each type of vertex is in the common /AMCPLC/, which is defined in AMPARM.

5) **Four momenta of external particles**
The four momenta of external particles are given in the arrays PEnnnn( ), where the fourth components correspond to the energies. An array PPROD(i,j) gives the inner products of particle momenta i and j. They are derived in KINEM and copied to these arrays in FUNC.

6) CHANEL **inputs for the external particles**
The arrays PSnnnn, EWnnnn, CEnnnn and EPnnnn are the lists of light-like vectors, weight factors, phase factors and list of polarization vectors, respectively, which are defined in section 2.4.

```
      PARAMETER (LOUTGO =  2, LINCOM =  1)
      PARAMETER (LANTIP = -1, LPRTCL =  1)
      PARAMETER (LSCALR =  1)
      PARAMETER (LEPEXA =  2, LEPEXW =  3, LEPEXZ =  3, LEPEXG =  2)
      PARAMETER (LEPINA =  4, LEPINW =  4, LEPINZ =  4, LEPING =  3)
      PARAMETER (LEXTRN =  2, LINTRN =  4)
* Table of amplitudes
      PARAMETER (NGRAPH =   28, NEXTRN =   5, LAG =  72)
      PARAMETER (NGRPSQ = NGRAPH*NGRAPH)
      COMMON /AMSLCT/JSELG(NGRAPH), JGRAPH, JHIGGS, JWEAKB
      COMPLEX*16 AG, APROP
      COMMON /AMGRPH/AG(0:LAG-1,NGRAPH), APROP(NGRAPH),
     &               ANCP(NGRAPH), ANSP(0:NGRAPH),
     &               CF(NGRAPH,NGRAPH), IGRAPH(NGRAPH)
```

Source list 3.3 Include file incl1.f

continue to the next page

```
* Masses and width of particles
      COMMON /AMMASS/AMWB,AMZB,AMAB,AMXB,AMX3,AMPH,AMLU,AMNE,AMNM,AMNT,
     &               . . . .
      COMMON /AMGMMA/AGWB,AGZB,AGAB,AGXB,AGX3,AGPH,AGLU,AGNE,AGNM,AGNT,
     &               . . . .
* Coupling constants
      COMMON /AMCPLC/CZWW       ,CAWW       ,CWWAA      ,CWWZA      ,
     &               . . . .
* Momenta of external particles
      COMMON /AMEXTR/PE0001(4),PE0002(4),PE0003(4),PE0004(4),
     &              PE0005(4),
     &              PPROD(NEXTRN, NEXTRN)
* Switch of gauge parameters
      COMMON /SMGAUS/IGAU00,IGAUAB,IGAUWB,IGAUZB,IGAUGL
      COMMON /SMGAUG/AGAUGE(0:4)
* Normalization
      COMMON /SMDBGG/FKNORM,FKCALL,NKCALL
* Calculated table of amplitudes
      COMMON /SMATBL/AV, LT, INDEXG
      COMPLEX*16 AV(0:LAG-1)
      INTEGER    LT(0:NEXTRN), INDEXG(NEXTRN)
* For external particles
      COMMON /SMEXTP/
     &    PS0001, EW0001, CE0001,
     &    PS0002, EW0002, CE0002,
     &    EP0003, EW0003,
     &    EP0004, EW0004,
     &    EP0005, EW0005
      REAL*8     PS0001(4,2), EW0001(1)
      COMPLEX*16 CE0001(2,2)
      REAL*8     PS0002(4,2), EW0002(1)
      COMPLEX*16 CE0002(2,2)
      REAL*8     EP0003(4,LEPEXW), EW0003(LEPEXW)
      REAL*8     EP0004(4,LEPEXW), EW0004(LEPEXW)
      REAL*8     EP0005(4,LEPEXA), EW0005(LEPEXA)
```

Source list 3.3 Include file `incl1.f`

3.2.2 Amplitude calculation

To calculate the numerical values of amplitudes, first the values of integration variables are translated into the four momenta of external particles, which is done by the subroutine `KINEM`. Then the subroutine `AMPTBL` is called to calculate the amplitudes.

Subroutine `AMPTBL`

The subroutine `AMPTBL` for the process $e^+e^- \to W^+W^-\gamma$ is shown in the source list 3.4, whose functions are as follows;

1) **External particles**
At the beginning of AMPTBL all the information about the external fermions and vector bosons are prepared in suitable form for the calculation of vertices as shown in the source list 3.4. For the external fermion (vector boson) the subroutine SMEXTF (SMEXTV) is called for this purpose, whose specifications are given in section 7.2.

```
      SUBROUTINE AMPTBL
** 5120     E+ E-  => W+  W-  A       TREE
      IMPLICIT REAL*8(A-H,O-Z)

      INCLUDE  'incl1.f'
      INCLUDE  'incl2.f'
*-----------------------------------------------------------
      JGRAPH = 0
*   External lines
      CALL SMEXTF(LINCOM,AMEL,PE0001,PS0001,CE0001)
      EW0001(1) = LPRTCL
      CALL SMEXTF(LOUTGO,AMEL,PE0002,PS0002,CE0002)
      EW0002(1) = LANTIP
      CALL SMEXTV(LEPEXW,AMWB,PE0003,EP0003,EW0003,IGAUWB)
      CALL SMEXTV(LEPEXW,AMWB,PE0004,EP0004,EW0004,IGAUWB)
      CALL SMEXTV(LEPEXA,AMAB,PE0005,EP0005,EW0005,IGAUAB)
* Graph NO.    1 -    1 (    1)
      IF (JSELG(    1).NE.0) THEN
      JGRAPH = JGRAPH + 1
      IGRAPH(JGRAPH) =    1
      CALL AM0001
      ENDIF
      . . . . .
      . . . . .

* Graph NO.   28 -    1 (   28)
      IF (JSELG(   28).NE.0) THEN
      JGRAPH = JGRAPH + 1
      IGRAPH(JGRAPH) =   28
      CALL AM0028
      ENDIF

      RETURN
      END
```

Source list 3.4 Example of subroutine AMPTBL

The variables LEPEXW and LEPEXA represent the spin freedoms of external W-bosons and photon, respectively, and are set in the include file incl1.f by the parameter statement as shown in the source list 3.3. For the fermion the variable EWnnnn(1) is set equal to “1” for particle or “-1” for anti-particle. In this example, `EW0001(1)` is set equal to "1" (electron) and `EW0002(1)` to "-1" (positron).

2) **Calculation of each amplitude**
The subroutine `AMnnnn` is called to calculate the `nnnn`-th graph. Since there are 28 graphs in the process $e^+e^- \to W^+W^-\gamma$, there are 28 subroutines from `AM0001` to `AM0028`. The flag `JSELG(`i`)` is used for selecting the graph. If it is set equal to "zero" in the subroutine `SETMAS` the corresponding i-th graph is not included in the calculation. This flag is to be set by the user for the time being, but it will be implemented in near future.

Subroutine `AMnnnn`

A main part of amplitude calculation appears in subroutines `AMnnnn`s. To describe the amplitude generation in section 2.4, we take a Feynman graph as an example in the process $e^+e^- \to W^+W^-\gamma$ shown in figure 2.1. The correspoding subroutine to the graph is `AM0026`, whose compositions are as follows;

1) **Internal momenta**
The internal momenta `PE0183` and `PE0185` are calculated from the external momenta, which correspond to those of internal neutrino and W-boson, respectively.

2) **Propagators**
The product of denominators of propagators is calculated by the subroutine `SMPRPD`, where the inputs are the momentum transfer, mass square and mass times width.

   The numerator of each propagator is handled by the subroutines `SMINTF` and `SMINTV` for the internal neutrino and W-boson, respectively.

3) **Vertices**
Numerical values of vertex amplitudes are calculated by subroutines `SMFFV` and `SMVVV`. By `SMFFV` the vertices $\bar{\nu}_e e^- W^+$ and $e^+\nu_e W^-$ are calculated and for the vertex $W^-W^+\gamma$ subroutine `SMVVV` is used. The calculated amplitudes of vertices $\bar{\nu}_e e^- W^+$ and $e^+\nu_e W^-$ and $W^-W^+\gamma$ are saved in the arrays `AV0122`, `AV0123` and `AV0124`, respectively.

4) **Connection of vertices**
First the vertices $\bar{\nu}_e e^- W^+$ and $e^+\nu_e W^-$ are connected by the routine `SMCONF`, where the amplitudes `AV0122` and `AV0123` are combined by summing over all the possible helicity states of the internal neutrino with weight `EW0183`. The resultant amplitude is stored in an array `AV0125`.

   Second the resultant amplitude `AV0125` and $W^-W^+\gamma$ amplitude `AV0124` are connected by taking summation over all the possible polarization states of internal W-boson with weight `EW0185` using the routine `SMCONV`. The total amplitude is saved in an array `AV`.

5) **Rearrange the internal structure of amplitude**
In order to sum up all amplitudes, they have to have the same internal structure. However, the internal structure of amplitude AV does strongly depend upon the order of constructing the amplitude, which may be different graph by graph. A subroutine AMPORD is used to change the amplitude AV in an individual structure into the amplitude AG in an unified structure.

```
* Graph No.   26 -   1 (  26)
************************************************************************
      SUBROUTINE AM0026
      IMPLICIT REAL*8(A-H,O-Z)

      INCLUDE 'incl1.f'
*-----------------------------------------------------------------------
      COMMON /AMWORK/
     &  PE0183, EW0183, PS0183, VM0183, CE0183,
     &  PE0185, EP0185, EW0185, VM0185,
     &  AV0122, AV0123, AV0124, AV0125
      COMMON /AMWORI/
     &  LT0122, LT0123, LT0124, LT0125
*         5856+         68 bytes used.
      REAL*8     PE0183(4), EW0183(2), PS0183(4,3), VM0183
      COMPLEX*16 CE0183(2,4)
      REAL*8     PE0185(4), EP0185(4,LEPINW), EW0185(LEPINW), VM0185
      INTEGER    LT0122(0:3)
      COMPLEX*16 AV0122(0:LINTRN*LEXTRN*LEPEXW-1)
      INTEGER    LT0123(0:3)
      COMPLEX*16 AV0123(0:LEXTRN*LINTRN*LEPINW-1)
      INTEGER    LT0124(0:3)
      COMPLEX*16 AV0124(0:LEPINW*LEPEXW*LEPEXA-1)
      INTEGER    LT0125(0:   4)
      COMPLEX*16 AV0125(0:LINTRN*LEPINW*LINTRN*LEPINW-1)
*-----------------------------------------------------------------------
*  Internal momenta
      DO   26 I = 1, 4
        PE0183(I) =  -PE0002(I) +PE0003(I)
        PE0185(I) =  -PE0004(I) -PE0005(I)
   26 CONTINUE
      APROP(JGRAPH) = 1.0D0
      VM0183 = - 2.0D0*PPROD(  2,  3) +  1.0D0*AMWB**2 +  1.0D0*AMEL**2
      CALL SMPRPD(APROP(JGRAPH),VM0183,AMNE**2,AMNE*AGNE)
      VM0185 = + 2.0D0*PPROD(  4,  5) +  1.0D0*AMWB**2 +  1.0D0*AMAB**2
      CALL SMPRPD(APROP(JGRAPH),VM0185,AMWB**2,AMWB*AGWB)
*   Internal lines
      CALL SMINTF(AMNE,PE0183,VM0183,EW0183,PS0183,CE0183)
      CALL SMINTV(LEPINW,AMWB,PE0185,EP0185,EW0185,VM0185,IGAUWB)
```

Source list 3.5 Subroutine AM0026 for $e^+e^- \to W^+W^-\gamma$

continue to the next page

```
*   Vertices
      CALL SMFFV(LINTRN,LEXTRN,LEPEXW,EW0183,EW0002,AMNE,AMEL,
     &           CWEL  (1,2),CE0183,CE0002,PS0183,PS0002,EP0003
     &           ,LT0122,AV0122)
      CALL SMFFV(LEXTRN,LINTRN,LEPINW,EW0001,EW0183,AMEL,AMNE,
     &           CWEL  (1,1),CE0001,CE0183,PS0001,PS0183,EP0185
     &           ,LT0123,AV0123)
      CALL SMVVV(LEPINW,LEPEXW,LEPEXA,-1,-1,-1,CAWW  ,PE0185,PE0004,
     &           PE0005,EP0185,EP0004,EP0005,LT0124,AV0124)
*   Connect vertices.
      CALL SMCONF(LT0123,LT0122,   2,   1,EW0183,AV0123,AV0122,
     &            LT0125,AV0125)
      CALL SMCONV(LT0124,LT0125,   1,   2,EW0185,AV0124,AV0125,
     &            LT,AV)
      APROP(JGRAPH) = +1.0D0/APROP(JGRAPH)
      INDEXG(   1) =    4
      INDEXG(   2) =    5
      INDEXG(   3) =    1
      INDEXG(   4) =    2
      INDEXG(   5) =    3
      CALL AMPORD(LT, AV, INDEXG, AG(0,JGRAPH))
      RETURN
      END
```

Source list 3.5 Subroutine AM0026 for $e^+e^- \to W^+W^-\gamma$

3.3 Specification of the kinematics routines

None of essential part of kinematics is generated by `GRACE`. The reason is that in general the choice of integration variables is highly dependent on the structure of singularities in the amplitude squared, such as infrared divergence, mass-singularity and t-channel photon exchange. It is quite difficult to prepare a kinematics enough general for any process. Therefore the user has to write the kinematics most appropriate for the process to be calculated. The subroutines which the user should complete are

`KINIT`	Initialization of kinematics.
`KINEM`	Calculate four-momenta of final particles from integration variables.

In the generated program the energy of a particle is stored in the 4-th component of the array which express the four-momentum.

3.3.1 Subroutine `KINIT`

`KINIT` makes the initialization of the kinematics and is called by `USERIN`. `BASES` calls `USERIN` at the beginning. The template of `USERIN` is generated by `GRACE` and is to be finalized by the user. The functions of `USERIN` are

1) **Initialization of Amplitude calculation**
   Call `SETMAS` and `AMPARM`.

2) **Initialization of kinematics**
   Call `KINIT` for intialization of kinematics

3) **Initialization of BASES parameters**
   Set the parameters for `BASES`. These parameters are transmitted to `BASES` through the commons `/BASE1/` and `/BASE2/`.

4) **Initialization of histograms**
   Let `BASES` know the number of histograms and that of scatter plots by calling `BHINIT` and initialize histograms.

5) **Initialization of amplitude summary table**
   After the integration a summary table of the contribution from each graph to the cross section is printed out. The buffer for the amplitude summary table is initialized.

An example of `USERIN` for the process $e^+e^- \to W^+W^-\gamma$ will be shown in section 3.5.3.

In the source list 3.6 the subroutine `KINIT` for the process $e^+e^- \to W^+W^-\gamma$ is shown, of which the program structure is as follows:

1) Masses `EM` and `WM` are set and are passed through the common `/MASS1/`.

2) Initialize the CM energy and parameters for kinematics and are passed through the commons ENRGY and TRNSF.

3) Initialize several cut parameters and are passed through the commons KCUTS and ACUTS.

4) The maximum number of multiplicity MXREG is set and are passed through the common /AMREG/. This parameter MXREG is used in FUNC (See section 3.5.4).

5) Finally the cut parameters are printed out.

```
      SUBROUTINE KINIT
      IMPLICIT REAL*8(A-H,O-Z)

* Masses and width of particles
      COMMON /AMMASS/AMWB,AMZB,AMAB,AMXB,AMX3,AMPH,AMLU,AMNE,AMNM,AMNT,
     &               AMLD,AMEL,AMMU,AMTA,AMQU,AMUQ,AMCQ,AMTQ,AMQD,AMDQ,
     &               AMSQ,AMBQ,AMCP,AMCM,AMCZ,AMCA,AMGL
      COMMON /AMGMMA/AGWB,AGZB,AGAB,AGXB,AGX3,AGPH,AGLU,AGNE,AGNM,AGNT,
     &               AGLD,AGEL,AGMU,AGTA,AGQU,AGUQ,AGCQ,AGTQ,AGQD,AGDQ,
     &               AGSQ,AGBQ,AGCP,AGCM,AGCZ,AGCA,AGGL

      COMMON / LOOP0 / LOOP
      COMMON / AMREG / MXREG
      COMMON / AMCNST/ PI, PI2, RAD, GEVPB, ALPHA
      COMMON / ENRGY / S,W,E,P,P1P2,FACT
      COMMON / TRNSF / YACO,EPSP,AP,XLOG
      COMMON / KCUTS / RMN,RMX,ETH
      COMMON / ACUTS / DELCUT,DLTCSG,DLTCS0,CSMX,CSMN
      COMMON / MASS1 / EM,WM
      COMMON / MASS2 / EM2,WM2

      DIMENSION WW(5)
      DATA WW / 200.D0, 300.D0, 400.D0, 400.D0, 1000.D0 /

*--------------------------- Entry point ------------------------------*
*---  1. Masses

            EM  = AMEL
            EM2 = EM*EM

            WM  = AMWB
            WM2 = WM*WM
*---  2. Initialize constants for kinematics.
C                                                    W  =  total energy
             W  =   WW( 5 )
             E  =   W/2.D0
```

Source list 3.6 KINIT for the process $e^+e^- \to W^+W^-\gamma$

continue to the next page

```
C-----------------------------------------------------  energy cuts
        ETH      =    WM
        IF(  ETH  .LT.  WM  )     ETH   =   WM
        RMN      =    1.D-3
        RMX      =    E - 4.D0*WM2/W
C-----------------------------------------------------  angle cuts
        CSMX     =    1.D0
        CSMN     =    - CSMX

        DLTCSG   =    0.D0
        DLTCSO   =    0.D0

        ZETAC    =    20.D0
        DELCUT   =    COS( ( 180.D0 - ZETAC )*RAD )
C------------------------------------------------------  energy variables
             P   =   SQRT(( E - EM )*( E + EM ))
             S   =   W*W
          P1P2   =   E*E + P*P
C-----------------------------------------  factors for initial raditaion
C                                           used in KINEM.
          EPSP   =   EM2/( P*( E + P ) )
            AP   =   EPSP + 1.D0
           ZZZ   =   ( 1.D0 - DLTCSG )/( DLTCSG + EPSP )
          XLOG   =   LOG( 1.D0 + 2.D0*ZZZ )
          YACO   =   XLOG/AP/P**2
C-----------------------------------------------  factor including flux.
          VREL   =   2.D0
          FLUX   =   VREL*S
          FACT   =   GEVPB/( FLUX*4.D0*(2.D0*PI)**4 )
*-----------------------------------------------------------------------
*--- 3. Set MXREG : the maximum number of values which are returned
*       by FUNC for one phase space point

      MXREG = 2
*-----------------------------------------------------------------------
      WRITE(6,100) W,EM
  100 FORMAT(1H1,///'   W = ',F8.2,3X,'  EM = ',G10.3,'    IN GEV')
      WRITE(6,'(''/  RMN ='',G10.3)') RMN
      WRITE(6,110) ETH,CSMX,CSMN
  110 FORMAT(1H ,///' (1) KINEMATICAL CUTS :'//
     &           10X,'      ETH = ',G10.3,'    GEV'/
     &           10X,'     CSMX = ',F8.3 ,         /
     &           10X,'     CSMN = ',F8.3 ,         //
     &           5X,' WHERE  ETH = THRESHOLD ENERGY FOR Q20 AND Q10'/
     &           5X,'       CSMN AND CSMX ARE ANGLE CUT FOR CS AND CSQ'
     &         //)

      RETURN
      END
```

Source list 3.6 KINIT for the process $e^+e^- \to W^+W^-\gamma$

3.3.2 Subroutine KINEM

In order to integrate the differential cross section, BASES samples a point in the integration volume and calls the function subprogram FUNC, which calculates the numerical value of integrand at the sampling point and returns it as the value of function. For calculating the differential cross section integration variables are to be translated into four-momenta of external particles, which is done by the subprogram KINEM. The outlook of KINEM is as follows:

```
      SUBROUTINE KINEM( NEXTRN, XX, PE, PP, YACOB, NREG, IREG, JUMP )

         IMPLICIT REAL*8
         PARAMETER (MXDIM = 50)
         INTEGER    NEXTRN,    NREG,         IREG,             JUMP
         REAL*8     XX(MXDIM), PE(4,NEXTRN), PP(NEXTRN,NEXTRN), YACOB
```

The meanings of the arguments are as follows;

NEXTRN	input	Number of external particles
XX	input	Values of integration variables at the sampling point
PE	output	Table of four momenta of external particles
PP	output	Table of inner products of four momenta
YACOB	output	Normalization for converting the square of amplitude to the cross section
NREG	in/out	The number of points in the phase space which correspond to a point in the integration volume. This value is set equal to one by FUNC for the first call at each sampling point.
IREG	input	Counter of calling KINEM at the same sampling point. Function FUNC increments IREG for each call, and calls KINEM while IREG $\leq$ NREG.
JUMP	output	If the sampling point is out of kinematical boundary, JUMP is set to be a non zero integer value.

An example of KINEM for the process $e^+e^- \to W^+W^-\gamma$ is shown in the source list 3.7. The specification for writing KINEM is as follows:

1) **Initialization**
   At the beginning of routine, variable JUMP should be cleared for the safety.

2) **Calculation of four vectors of external particles**
   From the integration variables XX(i), four vectors of external particles are derived and are stored in the arrays PE(1$\sim$4,k), where PE(1,k), PE(2,k) and PE(3,k) correspond to p_x, p_y and p_z, respectively, and PE(4,k) is energy of the k-th particle.

3) **Jump flag** `JUMP`
During calculation of four vector, when the sampling point `XX(`i`)` in the integration volume is out of the kinematical boundary, then the jump flag `JUMP` should be set equal to non-zero integer value. Otherwise, it must be zero.

4) **Inner products of four momenta**
The inner products, taking all combinations of the external four momenta, are calculated and stored in the array `PP(`l,m`)`, where the numbers l and m are corresponding to the labels of momenta `PE(1`$\sim$`4,`l`)` and `PE(1`$\sim$`4,`m`)`, respectively. Namely,

$$\begin{aligned}\mathtt{PP}(l,m) &= \mathtt{PE}(4,l)*\mathtt{PE}(4,m)-\mathtt{PE}(1,l)*\mathtt{PE}(1,m)\\ &\quad -\mathtt{PE}(2,l)*\mathtt{PE}(2,m)-\mathtt{PE}(3,l)*\mathtt{PE}(3,m).\end{aligned}$$

5) **Region flag** `NREG`
When the kinematics is represented by a many valued function, namely, a sampling point in the integration volume corresponds to several points in the momentum phase space, the number of multiplicity for this sampling point should be given as the value of `NREG`. The structure of `KINEM` for this case is as follows:

```
IF( IREG .EQ. 1 ) THEN

    The first call of KINEM for this sampling point.
    If this point corresponds to several points in the phase space,
    n points for example, then set NREG = n.

ELSE

    The IREG-th call for the same sampling point

ENDIF
```

In the example of 3.7 there is no such a structure, because the kinematics is constructed by a single valued function.

6) **Notice**
One should be careful not to loose the numerical accuracy by the cancellation over many digits which may take place when the inner-product `PP` are calculated from the four components of momenta. Use of invariants is recommended.

```
      SUBROUTINE KINEM(NEXTRN, XX, PE, PP, YACOB, NREG, IREG, JUMP)
      IMPLICIT REAL* 8(A-H,O-Z)

      PARAMETER ( MXDIM = 50 )
      INTEGER NEXTRN
      REAL*8  XX(MXDIM)
      REAL*8  PE(4,NEXTRN), PP(NEXTRN,NEXTRN)
      REAL*8  YACOB
      INTEGER NREG, IREG
      INTEGER JUMP

      COMMON /AMCNST/ PI, PI2, RAD, GEVPB, ALPHA
* Masses and width of particles
      COMMON /AMMASS/AMWB,AMZB,AMAB,AMXB,AMX3,AMPH,AMLU,AMNE,AMNM,AMNT,
     &               AMLD,AMEL,AMMU,AMTA,AMQU,AMUQ,AMCQ,AMTQ,AMQD,AMDQ,
     &               AMSQ,AMBQ,AMCP,AMCM,AMCZ,AMCA,AMGL
      COMMON /AMGMMA/AGWB,AGZB,AGAB,AGXB,AGX3,AGPH,AGLU,AGNE,AGNM,AGNT,
     &               AGLD,AGEL,AGMU,AGTA,AGQU,AGUQ,AGCQ,AGTQ,AGQD,AGDQ,
     &               AGSQ,AGBQ,AGCP,AGCM,AGCZ,AGCA,AGGL
*----------------------------------------------------------------------
      COMMON / ENRGY / S,W,E,P,P1P2,FACT
      COMMON / TRNSF / YACO,EPSP,AP,XLOG
      COMMON / KCUTS / RMN,RMX,ETH
      COMMON / ACUTS / DELCUT,DLTCSG,DLTCSO,CSMX,CSMN
      COMMON / MASS1 / EM,WM
      COMMON / MASS2 / EM2,WM2
*----------------------------------------------------------------------
      JUMP = 0
C--------------------------------------------------------- kinem-2
C                  Kinematics for the process
C
C            e-(P1) + e+(P2) ----> W-(Q1) + W+(Q2) + gamma(R)
C
C----------------------------------------------------------------------
C  (1)  Frame of reference :
C         (a)   Photon is along the z-axis.
C         (b)   Initial  e+  is in the x-z plane.
C  (2)  Definition of variables :
C         (a)   Polar angle of  e+  is  CSG.
C         (b)   Photon energy is  R.
C         (c)   Polar angle of  W+  is  CSO  and
C               azimuthal angle is  PHI.
C         (d)   Energies of  W-  and  W+  are  Q10 and Q20.
C         (e)   Angle between  e+  and  W+  is  CSTH.
C  (3)  Variable sequence :       R  --->  CSG  --->  Q20  --->  PHI
C----------------------------------------------------------------------
```

Source list 3.7 An example of KINEM

continue to the next page

```
C-----------------------------------------------------------------  R
          RR  =   RMX/RMN
           R  =   RMN*RR**XX(1)
          DR  =   LOG( RR )/R
C-----------------------------------------------------------------  CSG
         ZZZ  =   EXP( 2.D0*XLOG*( XX(2) - 0.5D0 ) )
         CSG  =   ( ZZZ - 1.D0 )/( ZZZ + 1.D0 )*AP
         SNG  =   SQRT(( 1.D0 - CSG )*( 1.D0 + CSG ))
         D2K  =   AP*( 2.D0/( 1.D0 + ZZZ ) )
         D1K  =   ZZZ*D2K
C------------------------------------------------------------- Q10, Q20
       CSOMX  =   1.D0 - DLTCS0
       RCS02  =   ( R*CSOMX )**2
          WR  =   W - R
          ER  =   E - R
           U  =   WR**2 - RCS02
           V  =   W*WR*ER
           D  =   RCS02*( ( W*ER )**2 - WM2*U )
         SQD  =   SQRT( D )
       Q20MX  =   ( V + SQD )/U
       Q20MN  =   ( V - SQD )/U
           A  =   E - Q20MX
           B  =   E - Q20MN
          CA  =   Q20MX - ER
          CB  =   Q20MN - ER
          RX  =   B*CA/( CB*A )
        DQ20  =   LOG( RX )/( S*R )
         ZZZ  =   A/CA*RX**XX(3)
         XXX  =   R*ZZZ/( 1.D0 + ZZZ )
         Q20  =   E - XXX
         Q2   =   SQRT(( Q20 - WM )*( Q20 + WM ))
         Q10  =   W - Q20 - R
         Q1   =   SQRT(( Q10 - WM )*( Q10 + WM ))
      IF(  Q20  .LT.  ETH   .OR.  Q10  .LT.  ETH   )     GOTO 9999
      IF(  CS01  .GT.  1.D0 - DLTCS0  )     GOTO 9999
C-----------------------------------------------------------------  CS0
         CS0  =   ( W*( E - R - Q20 ) + R*Q20 )/( Q2*R )
         SN0  =   SQRT(( 1.D0 - CS0 )*( 1.D0 + CS0 ))
C-----------------------------------------------------------------  PHI
        DPHI  =   2.D0*PI
       CSPHI  =   COS( DPHI*XX(4) )
       SNPHI  =   SIN( DPHI*XX(4) )
C-----------------------------------------------------------------  CSTH
        CSTH  =   CS0*CSG + SN0*SNG*CSPHI
      IF(  CSTH  .GT.  CSMX  .OR.  CSTH  .LT.  CSMN  )     GOTO 9999
```

Source list 3.7 An example of KINEM

continue to the next page

```
C----------------------------------------------------------------   CS01
              CS01  =   - ( R + Q2*CS0 )/Q1
C-----------------------------------------------------------------------
            CSQ  =   - ( R*CSG + Q2*CSTH )/Q1
      IF(  CSQ  .GT.  CSMX  .OR.  CSQ  .LT.  CSMN  )     GOTO 9999
         COSDEL  =   ( Q20*Q10 - W*( Q20 + Q10 )
     &                                   + E*W + WM2 )/( Q2*Q1 )
      IF(  COSDEL.  GT.  DELCUT  )     GOTO 9999
C----------------------------------------------------------  invariants
             D1  =   R*P*D1K
             D2  =   R*P*D2K
           EPSQ  =   WM2/( Q20 + Q2 )
             D3  =   R*E/ER*( EPSQ + Q2*( 1.D0 + CS0 ) )
             D4  =   R*( EPSQ + Q2*( 1.D0 - CS0 ) )
           P1Q2  =   E*Q20 + P*Q2*CSTH
           P2Q2  =   E*Q20 - P*Q2*CSTH
           Q1Q2  =   W*( E - R ) - WM2
           P1Q1  =   EM2 + P1P2 - D1 - P1Q2
           P2Q1  =   EM2 + P1P2 - D2 - P2Q2
*=======================================================================
*     Table of four momenta.
* PE(I, J) : I = 1 -> X, 2 -> Y, ... 4 -> energy, of J-th particle.

* 2:                                         EL+  INITIAL   LANTIP
      PE(1,2) = P*SNG
      PE(2,2) = 0.0D0
      PE(3,2) = P*CSG
      PE(4,2) = E
* 1:                                         EL-  INITIAL   LPRTCL
      PE(1,1) = -PE(1,2)
      PE(2,1) = -PE(2,2)
      PE(3,1) = -PE(3,2)
      PE(4,1) =  PE(4,2)
* 3:                                         WB+  FINAL     LPRTCL
      PE(1,3) = Q2*SN0*CSPHI
      PE(2,3) = Q2*SN0*SNPHI
      PE(3,3) = Q2*CS0
      PE(4,3) = Q20
* 5:                                         AB   FINAL     LPRTCL
      PE(1,5) = 0.0D0
      PE(2,5) = 0.0D0
      PE(3,5) = R
      PE(4,5) = R
* 4:                                         WB-  FINAL     LANTIP
      PE(1,4) = PE(1,1)+PE(1,2)-PE(1,5)-PE(1,3)
      PE(2,4) = PE(2,1)+PE(2,2)-PE(2,5)-PE(2,3)
      PE(3,4) = PE(3,1)+PE(3,2)-PE(3,5)-PE(3,3)
      PE(4,4) = Q10
```

Source list 3.7 An example of KINEM

continue to the next page

```
C---------------------------- momentum check :  (mass)**2 of particles

      ICHK = 0
      IF( ICHK .NE. 0 ) THEN
          PRINT *, '-----------------------'
          PRINT *, ' EM**2, WM**2,  ', EM2, WM2
          PRINT *, '-----------------------'
          DO 10 J = 1,5
             SQP = PE(1,J)**2+PE(2,J)**2+PE(3,J)**2
   10        PRINT *, ' mass**2   =    ', PE(4,J)**2 - SQP
      ENDIF

C---- PP(I,J) = inner product between PE(*,I) and PE(*,J) --  invariants

      PP(1,1) = EM2
      PP(1,2) = P1P2
      PP(1,3) = P1Q2
      PP(1,4) = P1Q1
      PP(1,5) = D1
      PP(2,1) = P1P2
      PP(2,2) = EM2
      PP(2,3) = P2Q2
      PP(2,4) = P2Q1
      PP(2,5) = D2
      PP(3,1) = P1Q2
      PP(3,2) = P2Q2
      PP(3,3) = WM2
      PP(3,4) = Q1Q2
      PP(3,5) = D4
      PP(4,1) = P1Q1
      PP(4,2) = P2Q1
      PP(4,3) = Q1Q2
      PP(4,4) = WM2
      PP(4,5) = D3
      PP(5,1) = D1
      PP(5,2) = D2
      PP(5,3) = D4
      PP(5,4) = D3
      PP(5,5) = 0.0D0

C-----------------------------------------------------------  Jacobian
      YACOB  =   FACT*DR*(YACO*D1*D2)*(DQ20*D3*D4)*DPHI/2.D0

      RETURN
 9999 JUMP = 1
      RETURN
      END
```

Source list 3.7 An example of `KINEM`

3.4 Test of generated source code

The main program to check gauge invariance at one point in the integration volume is produced by `GRACE`, shown in source list 3.8. Before running this program the full set of program components for the integration, namely `USERIN`, `KINIT`, `FUNC`, and `KINEM`, should be prepared. The program flow of this test is as follows:

1) **Initialization**
   Call `USERIN` to initialize the parameters for calculating the differential cross section with `FUNC`.

2) **Select a point**
   A point is selected in the integration volume. When one wants to test by several different points, take the following structure:

```
      PARAMETER (NPOINT = 5)
      REAL*8 XX(NPOINT)
      DATA XX / 0.1, 0.3, 0.5, 0.7, 0.9 /
      . . . . .

      CALL USERIN
      DO 20 K = 1, NPOINT
         (1) calculate the function with the unitary gauge
             and print out the result
         (2) calculate the function with the covariant gauge
             and print out the result
   20 CONTINUE
```

   If the kinematics has an experimental cut which does not include the selected point, one cannot make the check correctly. It is recommended to make the check for various points in the integration volume.

3) **Calculation in the unitary gauge**
   The differential cross section is calculated in the unitary gauge and the result is printed out.

4) **Calculation in the covariant gauge**
   The differential cross section is calculated in the covariant gauge and the result is printed out.

The subprograms `USERIN` and `FUNC` call the histogram package. Since, however, this test program calculates the integrand at a point in the integration volume, the histogram has no meaning. Thus we use just dummy routines of the histogram package so that we do not need to change the subprograms `USERIN` and `FUNC` at all.

```
* Test main program
      IMPLICIT REAL*8(A-H,O-Z)

      PARAMETER ( MXDIM = 50 )
      COMMON / LOOP0 / LOOP
      COMMON / BASE1 / XL(MXDIM),XU(MXDIM),NDIM,NWILD,
     &                 IG(MXDIM),NCALL
      COMMON / BASE2 / ACC1,ACC2,ITMX1,ITMX2
      COMMON / BASE3 / SI,SI2,SWGT,SCHI,SCALLS,ATACC,NSU,IT,WGT
*-----------------------------------------------------------------------
      DIMENSION X(MXDIM)
      INCLUDE 'incl1.f'
      INCLUDE 'incl2.f'
*-----------------------------------------------------------------------
      WRITE(*,'(10X,A//)')'* 5120     E+ E-  => W+  W-  A       TREE '

      CALL USERIN
      WGT = 1.D0
CT    DO 20 MANY X'S

      DO 10 I = 1, NDIM
        X(I) = 0.45D0
   10 CONTINUE
      DO 40 I = 1, NGRAPH
        JSELG(I) = 1
   40 CONTINUE

      WRITE(*,*) 'X     = ', (X(I),I=1,NDIM)
      WRITE(*,*) 'JSELG = ', JSELG

*   Unitary gauge
      IGAUAB = 0
      IGAUWB = 0
      IGAUZB = 0
      IGAUGL = 0

      AGAUGE(0) = 1.0D20

      WRITE(*,*) 'IGAUAB = ', IGAUAB, ' AGAUGE = ',AGAUGE(IGAUAB)
      WRITE(*,*) 'IGAUWB = ', IGAUWB, ' AGAUGE = ',AGAUGE(IGAUWB)
      WRITE(*,*) 'IGAUZB = ', IGAUZB, ' AGAUGE = ',AGAUGE(IGAUZB)
      WRITE(*,*) 'IGAUGL = ', IGAUGL, ' AGAUGE = ',AGAUGE(IGAUGL)

      AAA = FUNC(X)
      WRITE(*,*) '  ANS1   = ',AAA
      WRITE(*,*) '# GRAPHS = ',JGRAPH
```

Source list 3.8 Main program for gauge invariance check

continue to the next page

```
*   Covariant gauge
      IGAUAB = 1
      IGAUWB = 2
      IGAUZB = 3
      IGAUGL = 4
      AGAUGE(IGAUAB) = 2.0D0
      AGAUGE(IGAUWB) = 3.0D0
      AGAUGE(IGAUZB) = 4.0D0
      AGAUGE(IGAUGL) = 5.0D0

      WRITE(*,*) 'X     = ', (X(I),I=1,NDIM)
      WRITE(*,*) 'JSELG = ', JSELG

      WRITE(*,*) 'IGAUAB = ', IGAUAB, ' AGAUGE = ',AGAUGE(IGAUAB)
      WRITE(*,*) 'IGAUWB = ', IGAUWB, ' AGAUGE = ',AGAUGE(IGAUWB)
      WRITE(*,*) 'IGAUZB = ', IGAUZB, ' AGAUGE = ',AGAUGE(IGAUZB)
      WRITE(*,*) 'IGAUGL = ', IGAUGL, ' AGAUGE = ',AGAUGE(IGAUGL)

      BBB = FUNC(X)
      WRITE(*,*) 'ANS2     = ',BBB
      WRITE(*,*) '# GRAPHS = ',JGRAPH

      IF(BBB.NE.0) THEN
        WRITE(*,*) 'ANS1/ANS2 - 1 = ', AAA/BBB - 1
      ELSE
        WRITE(*,*) 'ANS1 = ', AAA,'  ANS2 =', BBB
      ENDIF
CT 20 CONTINUE

      STOP
      END
```

Source list 3.8 Main program for gauge invariance check

In the output 3.1 the squared values of the amplitude at a point in the phase space in different gauges, covariant gauge and unitary gauge are shown, which is the output of the test program. The number of calculated Feynman graphs is different in these gauges. The relative error is printed in the output. We usually require about 14 digits agreement in double precision and about 32 digits in quadruple precision [1]. In the output one can also see contribution of each graph to the result.

[1]In quadruple precision, we have checked it on FACOM mainframe computer, Sun sparc workstation and HITAC 3050 workstation

```
                  GRACE   Ver.   1.0


        * 5120     E+ E-  => W+  W-  A       TREE


  W =  1000.00     EM =   .511E-03    IN GEV
  RMN =  .100E-02


(1) KINEMATICAL CUTS :

              ETH =   80.0        GEV
             CSMX =    1.000
             CSMN =   -1.000

     WHERE  ETH = THRESHOLD ENERGY FOR Q20 AND Q10
            CSMN AND CSMX ARE ANGLE CUT FOR CS AND CSQ


X     =  .45 .45 .45 .45
JSELG =  1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
IGAUAB =  0 AGAUGE =  1.000000000000000E+20
IGAUWB =  0 AGAUGE =  1.000000000000000E+20
IGAUZB =  0 AGAUGE =  1.000000000000000E+20
IGAUGL =  0 AGAUGE =  1.000000000000000E+20
  ANS1   =  1.34979414428365
# GRAPHS =  18
X     =  .45 .45 .45 .45
JSELG =  1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
IGAUAB =  1 AGAUGE =  2.0
IGAUWB =  2 AGAUGE =  3.0
IGAUZB =  3 AGAUGE =  4.0
IGAUGL =  4 AGAUGE =  5.0
ANS2     =  1.34979414428364
# GRAPHS =  28
ANS1/ANS2 - 1 =  5.551115123125782E-15

INTEGRATED VALUE OF SQUARE OF EACH GRAPH
GRAPH          ABSOLUTE         RELATIVE
    1 :     .23935707E-04     .17732857E-04
    2 :     .29254454E-04     .21673271E-04
    3 :     .30405158E-10     .22525774E-10
    4 :     .24923917E+02     .18464976E+02
    5 :     .30462254E+02     .22568074E+02
    6 :     .16871906E-06     .12499614E-06
    7 :     .33536397E+02     .24845564E+02
    8 :     .40988511E+02     .30366490E+02
    9 :     .14039649E+01     .10401326E+01
   10 :     .11486965E+01     .85101606E+00
   11 :     .34248206E+02     .25372910E+02
   12 :     .50292151E+03     .37259127E+03
   13 :     .37723272E+03     .27947426E+03
   14 :     .18116921E-07     .13421988E-07
```

Output 3.1 Result from gauge invariance check

continue to the next page

```
   15 :     .30004545E+02     .22228978E+02
   16 :     .82173348E+02     .60878430E+02
   17 :     .42105481E+01     .31194002E+01
   18 :     .92509178E+03     .68535767E+03
   19 :     .14010416E-11     .10379669E-11
   20 :     .46106302E+03     .34158025E+03
   21 :     .37723271E+03     .27947426E+03
   22 :     .46034419E-09     .34104770E-09
   23 :     .10452751E-13     .77439594E-14
   24 :     .31070077E+02     .23018382E+02
   25 :     .10159998E-13     .75270725E-14
   26 :     .83905907E+02     .62162002E+02
   27 :     .28065832E+01     .20792676E+01
   28 :     .92394308E+03     .68450666E+03
TOTAL :     .13497941E+01
```

Output 3.1 Result from the gauge invariance check

3.5 Numerical integration

The GRACE system generates a set of FORTRAN subprograms necessary for the Monte Carlo integration program package BASES, which consists of MAINBS, USERIN, FUNC and USROUT. In this section a description of the integration program package BASES and these subprograms generated by GRACE are given in the following order:

(1) **Job parameters**
In the initialization stage of integration, BASES reads these parameters to control the program flow.

(2) **Program structure of** BASES
Relation among BASES and those subprograms generated by GRACE is described in section 3.5.2.

(3) **Initialization subprogram** USERIN in section 3.5.3.

(4) **Function program of integrand** FUNC in section 3.5.4.

(5) **Histogram package** in 3.5.5.

(6) **Output from** BASES in 3.5.6.

3.5.1 Job parameters

In order to control the integration job, there are four job parameters; loop parameters, print flag, computing time limit and job flag. They are read at the beginning of a job on a *main frame computer*, while they are to be given interactively from a terminal on a *unix machine*.

Job flag

The integration may take much computer time, for instance, if calculation of the integrand needs a long computation. Then it might happen **on a main frame computer** that a job is terminated before reaching the convergence condition of integration due to the computing time limit of the job class and lose all information in the worst case. To prevent this trouble, BASES watchs the remaining computing time and when it is not enough for the next iteration, all temporary information is saved on a disc file before the job is terminated. The meaning of job flag is as follows;

JFLAG	meaning of job flag
0	First trial of the grid optimization step
1	First trial of the integration step
2	Continuation of the grid optimization step
3	Continuation of the integration step.

At the beginning of a new integration job, JFLAG = 0 should be set. If the job is terminated for lack of the computing time, the value of next job flag is given at the end of the output.

If we ask better accuracy or more iterations than the present result, by setting JFLAG = 3, *further continuation* of the integration step can be carried out *even once after the integration finishes by achieving a given accuracy of the estimate or reaching a given number of iterations.* In order to continue the integration step further, the maximum iteration number ITMX2 and the expected accuracy ACC2 are to be set larger and smaller than the previous ones, respectively, in addition to set JFLAG = 3.

To use this option, a file should be prepared beforehand, which we call the probability information file (see section 3.5.6). On a main frame computer this file is allocated to the logical unit number 23 in the JCL (see section 4.2.5).

Since **on a unix machine** the computing time limit is not usually settled, a job will run until the convergence condition is achieved. Therefore *only* JFLAG = 0 *and* 3 *are meaningful on a unix machine.* However, if user wants to generate four vectors by SPRING, the probability information file should be prepared in the integration stage. For this case, it should be opened with the logical unit number 23 in the main program MAINBS.

Loop parameters

When we want to know the energy dependence of the cross section for instance, we must integrate the differential cross section at several energy points. To make this possible in a single job, the loop option is prepared. To use the loop option,

(a) insert statements like the following in subprogram USERIN;

```
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 )
     ..............
```

In this example the CM energy W is selected from six energy points WCM(6) by the number LOOP, which is counted by BASES.

(b) set the loop parameters. The loop parameters consist of the first loop number LOOPF and last loop number LOOPL of the loop.

When the loop parameters are given as (LOOPF, LOOPL) = (4, 4), then only the fourth energy point W = 105.0 is calculated. If (LOOPF, LOOPL) = (1, 3) are given, three energy points are selected successively from W = 40.0 to 70.0.

Print flag

Since there are several kinds of outputs from BASES, selecting the output by the print flag is useful. The outputs from BASES are the following:

(a) final result of the integration.

(b) convergence behavior for the grid optimization step,

(c) histograms and scatter plots for the grid optimization step,

(d) convergence behavior for the integration step, and

(e) histograms and scatter plots for the integration step

A combination of the above outputs is printed according to the absolute value of the print flag, which defined as follows:

|NPRINT|	meaning of print flag
0	nothing but USROUT is called
1	only (a)
2	(a) and (e)
3	(d) and (e)
4	(b), (d) and (e)
≥ 5	(b), (c), (d) and (e)

If the *negative* number "$-$NPRINT" is given as the print flag, the user output routine USROUT is called at the end of the job as well as one of the above combinations is printed according to NPRINT. The routine USROUT should be prepared by user if the negative number or "0" is specified as the print flag.

Computing time limit

(*This facility functions only for a main frame computer.*) The computing time limit is to be given as a job parameter in the unit of minute with real number, while the internal time in BASES counts in the unit of second. When the remaining computing time is not enough for the next iteration, the job is terminated (see also the item of job flag).

3.5.2 Program structure of BASES

In figure 3.4 is shown the structure of program BASES, where MAINBS, USERIN, KINIT, KINEM and FUNC are generated by GRACE and are to be finalized by user.

Program flow

The main program MAINBS calls the steering routine BSMAIN, which controls the program flow of integration as follows:

Fig. 3.4 Program structure of BASES

(A) Initialization

 (1) At the beginning, the job parameters described in the previous section are read from the unit number 5.
  On a *unix system*, these parameters are obtained from terminal by subprogram `BSUNIX`.
 (2) In subprogram `BSUSRI`, the parameters for `BASES` are set to the default values and the subprogram `USERIN` is called to initialize them. If some fundamental parameters, like number of dimensions of integral, are not set in `USERIN`, the program will stop.
  Specification of `USERIN` is given in section 3.5.3.
 (3) If the job flag is *not* equal to "0", then the current results are read from the file by the subprogram `BSREAD`.
 (4) If the job flag is equal to "0", then the widths of all grids are set uniform.

(B) The grid optimization and integration steps

 (1) For each hypercube, N_{trial} points are sampled in the following way;
  (a) Sample a small region in the hypercube and sample a point in the small region for each variable.
  (b) Call function subprogram `FUNC` to calculate the differential cross section at the sampled point.

  and the estimate and variance of the integral are calculated.
 (2) Sums of the estimates and variances over all hypercubes are taken to calculate the estimate and error of the integral.
 (3) If the integration converges, then go to step (C).
 (4) If the integration does not yet converge, then;

  For the grid optimization step,
   call the subprogram `BSGDEF` to adjust each width of grids and then go to step (B.1).

  For the integration step,
   go to step (B.1).

(C) Termination of task

 (1) Print out the result.
 (2) When this is the grid optimization step, set the job flag equal to "1" and go to step (B.1).
 (3) When this is the integration step, call `USROUT`, write the probability information on a file and stop.

Main program `MAINBS`

In the program `MAINBS`, the name of function program should be given by an external statement and subprogram `BSMAIN` is called, which is a steering routine of the integration. Furthermore, to secure the histogram buffer the common /PLOTB/ is declared in this main. An example of MAINBS for the *unix system* is given in the source list 3.9.

```
      EXTERNAL FUNC
      DOUBLE PRECISION FUNC

      INCLUDE 'inclh.f'
      COMMON /PLOTB/ IBUF(281*NHIST + 2527*NSCAT + 281)

      open( 23, file ='bases.data',status='unknown',form='unformatted')

      CALL BSMAIN( FUNC )

      close(23)

      STOP
      END
```

Source list 3.9 An Example of MAINBS

The file inclh.f is referred by the INCLUDE statement, where the number of histograms and that of scatter plots are given by the PARAMETER statement as follows:

```
      PARAMETER ( NHIST = 5, NSCAT = 6 )
```

The reason why we use the include file to define the numbers of histograms and scatter plots is that it is much better to change them in a include file than to change the all subprograms including them since these numbers are used in several subprograms.

A binary file bases.data is created for the probability information as mentioned in previous section.

If the name of function program differs from FUNC, it should be declared with the real name instead of FUNC.

3.5.3 Initialization subprogram USERIN

At the beginning of the integration job, the subroutine USERIN is called to initialize the parameters both for the integration and calculating the integrand. The template of USERIN is generated by GRACE and is to be finalized by the user. The functions of USERIN are as follows:

(1) **Initialization of the amplitude calculation**
This is done by calling subprograms SETMAS and AMPARM,which are described in section 3.2.1.

(2) **Initialization of kinematics**

This is done by calling the subprogram KINIT. KINIT must be prepared by the user, specification of which is given in section 3.3.

(3) **Initialization of the integration parameters**

The parameters for integration are summarized in the commons /BASE1/ and /BASE2/, where all real variables are to be given by the double precision.

```
PARAMETER ( MXDIM = 50 )
COMMON /BASE1/ XL(MXDIM), XU(MXDIM), NDIM ,NWILD, IG(MXDIM), NCALL
```

XL(i) (i = 1, NDIM)	The lower bound of i-th variable.
XU(i) (i = 1, NDIM)	The upper bound of i-th variable.
NDIM	The dimension of the integral.
NWILD	The number of wild variables (at least one and at most 15).
IG(i) (i = 1, NDIM)	The flag for the grid optimization. If IG(i) $= 1(0)$, the grid for the i-th variable is (not) optimized. If the integrand is approximately constant for a variable, it may give better convergence than varying widths to set the grid uniform for this variable. The default flag is "1" (optimization).
NCALL	The number of sampling points per iteration.

The number of real sampling points differs from a given number NCALL, which is automatically determined by the following algorithm. It is noticed that the order of variables XU(i), XL(i) and IG(i) ($i = 1$, NDIM), should start with the wild variables.

$N_{call}^{(given)}$	$=$	1,000			$N_{call}^{(given)}$	$=$	5,000		
N_{wild}	N_s	N_{cube}	N_g	$N_{call}^{(real)}$	N_{wild}	N_s	N_{cube}	N_g	$N_{call}^{(real)}$
1	25	25	50	1,000	1	25	25	50	5,000
2	22	484	44	968	2	25	625	50	5,000
3	7	343	49	686	3	13	2,197	39	4,394
4	4	256	48	768	4	7	2,401	49	4,802
5	3	243	48	972	5	4	1,024	48	4,096
6	2	64	50	960	6	3	729	48	4,374
7	2	128	50	896	7	3	2,187	48	4,374
8	2	256	50	768	8	2	256	50	4,864
9	1	1	50	1,000	9	2	512	50	4,608
10	1	1	50	1,000	10	2	1024	50	4,096
$N_{call}^{(given)}$	$=$	10,000			$N_{call}^{(given)}$	$=$	20,000		
N_{wild}	N_s	N_{cube}	N_g	$N_{call}^{(real)}$	N_{wild}	N_s	N_{cube}	N_g	$N_{call}^{(real)}$
1	25	25	50	10,000	1	25	25	50	20,000
2	25	625	50	10,000	2	25	625	50	20,000
3	17	4,913	34	9,826	3	21	9,261	42	18,522
4	8	4,096	48	9,182	4	10	10,000	50	20,000
5	5	3,125	50	9,375	5	6	7,776	48	15,552
6	4	4,096	48	8,192	6	4	4,096	48	16,384
7	3	2,187	48	8,448	7	3	2,187	48	19,693
8	2	256	50	9,984	8	3	6,561	48	19,683
9	2	512	50	9,728	9	2	512	50	19,968
10	2	1,024	50	9,216	10	2	1,024	50	19,456

The number of subregions per variable N_s is determined by the maximum number which satisfies the two inequalities:

$$N_s = (\frac{N_{call}}{2})^{\frac{1}{N_{wild}}} \leq 25 \quad \text{and} \quad N_s^{N_{wild}} < 32768.$$

The number of hypercubes is given by $N_{cube} = N_s^{N_{wild}}$, then the number of sampling points per hypercube is $N_{trial} = N_{call}/N_{cube}$. Since the number N_{trial} is an integer, the calculated number $N_{call}^{(real)} = N_{trial} \times N_{cube}$ may differ from the given number $N_{call}^{(given)}$.

The table gives the numbers of real sampling points $N_{call}^{(real)}$ depending on the given numbers of sampling points $N_{call}^{(given)}$ and the numbers of wild variables N_{wild}.

```
        COMMON /BASE2/ ACC1, ACC2, ITMX1, ITMX2
```

`ACC1`	The accuracy (%) for the grid optimization step (default 0.2 %).
`ACC2`	The accuracy (%) for the integration step (default 0.05 %).
`ITMX1`	The maximum iteration number of the grid optimization step (default 15).
`ITMX2`	The maximum iteration number of the integration step (default 100).

(4) **Numbers of Histograms and Scatter plots**
In order to let the system know the numbers of histograms and scatter plots, the subroutine `BHINIT` is to be called somewhere in `USERIN` like;

```
      CALL BHINIT( NHIST, NSCAT )
```

where `NHIST (NSCAT)` is the number of histograms (scatter plots) and one histogram (scatter plot) requires 281 (2527) 32-bit words. An additional storage of 281 words is kept for a histogram of the numbers of trials in `SPRING`. It is noted that the buffer for the histograms and scatter plots should be secured in the main program `MAINBS`.

(5) **Initialization of Histograms and Scatter plots**
To make a histogram and a scatter plot, the following initialization routines are to be called in the `USERIN`;

```
    CALL XHINIT( ID#,
  .              lower_limit, upper_limit, # of bins, ' Title '),
and
```

```
 CALL DHINIT( ID#,
.              x_lower_limit, x_upper_limit, # of x bins,
.              y_lower_limit, y_upper_limit, # of y bins,
.                                                ' Title '),
```

respectively. The ID and bin numbers are to be given by an integer value, and the lower and upper limits are to be given by the double precision values. The maximum bin number both for histograms and scatter plots is 50, which is defined by the paper size. When too many histograms or scatter plots are initialized, the first NHIST-1 histograms and NSCAT scatter plots are initialized and the others are neglected.

An example for the process $e^+e^- \to W^+W^-\gamma$ is shown in the source list 3.10. In this example, the histograms for all integration variables and scatter plots for all combinations of integration variables are demanded, which is a standard set of histograms and scatter plots demanded in the generated USERIN by GRACE. For this case, the parameters NHIST and NSCAT are to be set equal to at least NDIM and NDIM*(NDIM-1)/2 in the include file inclh.f, respectively.

```
      SUBROUTINE USERIN
      IMPLICIT REAL*8(A-H,O-Z)

      PARAMETER ( MXDIM = 50 )
      COMMON / LOOP0 / LOOP
      COMMON / BASE1 / XL(MXDIM),XU(MXDIM),NDIM,NWILD,
     &                 IG(MXDIM),NCALL
      COMMON / BASE2 / ACC1,ACC2,ITMX1,ITMX2
      COMMON / BASE3 / SI,SI2,SWGT,SCHI,SCALLS,ATACC,NSU,IT,WGT
* Table of amplitudes
      PARAMETER (NGRAPH =   28, NEXTRN =   5, LAG =  72)
      PARAMETER (NGRPSQ = NGRAPH*NGRAPH)
      COMMON /AMSLCT/JSELG(NGRAPH), JGRAPH, JHIGGS, JWEAKB
      COMPLEX*16 AG, APROP
      COMMON /AMGRPH/AG(0:LAG-1,NGRAPH), APROP(NGRAPH),
     &               ANCP(NGRAPH), ANSP(0:NGRAPH),
     &               CF(NGRAPH,NGRAPH), IGRAPH(NGRAPH)

      INCLUDE 'inclh.f'
      CHARACTER XSTR*14, DSTR*24
*=====================================================================
*          Parameters for Amplitude calculation
*=====================================================================
```

Source list 3.10 An example of USERIN

continue to the next page

```
*         ============= Mass and Width
           CALL SETMAS
*         =============
*         ============= Coupling constants
           CALL AMPARM
*         =============
*=================================================================
*          Initialization of Kinematics
*=================================================================
*         ============
           CALL KINIT
*         ============
*=================================================================
*          Parameters for BASES
*=================================================================
*--- 1. Dimension of integration.

           NDIM  = 4
           NWILD = 4

*--- 2. Region of integration.

   DO 10  I = 1, NDIM
              XL(I) = 0.D0
              XU(I) = 1.D0
      IG(I) = 1
   10      CONTINUE

*--- 3. Numbers of iterations and sampling points / iteration
        and expected accuracies

           ITMX1  =  5
           ITMX2  =  5

           ACC1   = 0.2D0
           ACC2   = 0.01D0

           NCALL  = 5000
*=================================================================
*          Initialization of Histograms and scatter plots
*=================================================================
*     Change NHIST = NDIM, NSCAT = NDIM*(NDIM-1)/2
*     in the parameter statement ( in INCLH )

          CALL BHINIT(NHIST,NSCAT)
```

Source list 3.10 An example of `USERIN`

continue to the next page

```
         NX = 50
         ND = 50

         DO 100 I = 1, NDIM
            WRITE(XSTR, 110) I
 110        FORMAT('X(',I2,') Spectrum')
            CALL XHINIT(I, XL(I), XU(I), NX, XSTR)
 100     CONTINUE

         K = 0
         DO 200 I = 1, NDIM - 1
         DO 200 J = I + 1, NDIM
            WRITE(DSTR, 210) I, J
 210        FORMAT('X(',I2,')-X(',I2,') Distribution')
            K = K + 1
            CALL DHINIT(K, XL(I),XU(I),ND, XL(J),XU(J),ND,DSTR)
 200     CONTINUE

*======================================================================
*          Initialization of summary table
*======================================================================

         DO 300 IGR = 0, NGRAPH
           ANSP(IGR) = 0.0D0
 300     CONTINUE
         FKCALL = 0
         NKCALL = 0

      RETURN
      END
```

Source list 3.10 An example of USERIN

3.5.4 Function program of the integrand

The function program calculates the value of integrand at the sampling point fed by BASES.

A set of numerical values of the integration variables at a sampling point is passed through the argument of function program. A typical structure of the function program is given in the source list 3.11 where the dimension of integration is five. A recipe for writing the function program is as follows:

1) Calculate the kinematical variables, by which the differential cross section is described, from the integration variables, X(i) for $i = 1$, NDIM.

2) If, in the last step, a sampling point is found to be outside of the kinematical boundary, set the value of function equal to zero and return.

3) If the point is inside the kinematical boundary, calculate the numerical value of the differential cross section and set the value of function equal to the calculated value.

4) If a histogram is required, call subprogram XHFILL once.

5) If a scatter plot is required, call subprogram DHFILL once.

```
      DOUBLE PRECISION FUNCTION FUNC(X)
      DOUBLE PRECISION X(5)
      FUNC = 0.0

      ... Calculation of the kinematics ...
      IF( the point is outside the kinematical boundary ) RETURN

      FUNC = is calculated from X(i) for i = 1, 5.

      CALL XHFILL( ID, V, FUNC )
      CALL DHFILL( ID, VX, VY, FUNC)

      RETURN
      END
```

Source list 3.11 Typical structure of FUNC

An example of FUNC for the process $e^+e^- \to W^+W^-\gamma$ is given in the source list 3.12. The structure of this example is as follows:

1) The array XX stores the values of integration variables.

2) Total number of external particles NEXTRN and XX are transferred to subprogram KINEM. The tables of momenta P and inner-products of them PP, and normalization factor YACOB are received from KINEM.
In the case of QCD calculation, the running coupling constant can be included in YACOB, which should be defined by the user.

3) P and PP are copied to the common variables PExxxx and PPROD, respectively, and they are used in the amplitude calculation.

4) Subprogram AMPTBL calculates the amplitudes and makes the tables of them.

5) Summation over the spin states by calling the subprogram AMPSUM.

6) Fill the histograms and scatter pots by the subprograms XHFILL and DHFILL, respectively.

7) The variable `JUMP`

   If the sampling point is out of the kinematical boundary, `JUMP` is set to a non zero integer in `KINEM`. For this case, the amplitude does not need to be calculated.

8) The variables `NREG` and `IREG`

   When the kinematics contains a multi-valued function, *i.e.* one sampling point in the integration volume corresponds to several points in the phase space, the variables `NREG` and `IREG` take the total number of multiplicity and the current number of multiplicity, respectively.

   In the subprogram `FUNC`, the variables `NREG` and `IREG` are set to "1" at the beginning and subroutine `KINEM` is called. A typical structure of subroutine `KINEM` for multi-valued function is as following:

```
SUBROUTINE KINEM(NEXTERN, XX, P, PP, YACOB, NREG, IREG, JUMP)
...
IF(IREG.EQ.1) THEN
    NREG = (the number of multiplicity at the sampling point XX)
    (Calculate four momenta P for the first calculation)
    (Calculate inner products of four momenta PP)
    (Calculate Jacobian YACOB for the first calculation)
    ...
ELSE IF(IREG.EQ.2) THEN
    (Calculate four momenta P for the second calculation)
    (Calculate inner products of four momenta PP)
    (Calculate Jacobian YACOB for the first calculation)
    ...
ELSE IF ...
    ...
ENDIF
RETURN
END
```

   `KINEM` calculates the total number of multiplicity at the sampling point and store it in `NREG`. If it is greater than "1", then the first calculation of four momenta is performed. From the second calculation, `IREG` is incremented with keeping `NREG` unchanged and momenta are returned by calling `KINEM`. The same step is repeated until `IREG` reaches `NREG`. The value of `MXREG` is defined in the subroutine `KINIT` and is used to protect unexpected repeat. It is clear that `NREG` is the total number of multiplicity at a sampling point given by `KINEM` and `IREG` plays the role of counter which shows the number of `KINEM` calls.

9) When one wants to demand some experimental cut on the phase space, one can define it either in `KINEM` or in a new subroutine. The new subroutine should be called just after calling `KINEM`.

If the sampling point falls into the region excluded by the experimental cut, then

```
GO TO 1000
```

is executed.

```
      FUNCTION FUNC(X)
      IMPLICIT REAL*8(A-H,O-Z)
      PARAMETER ( MXDIM = 50 )
      REAL*8     FUNC
      REAL*8   X(MXDIM)

      COMMON / LOOP0 / LOOP
      COMMON / BASE1 / XL(MXDIM),XU(MXDIM),NDIM,NWILD,
     &                 IG(MXDIM),NCALL
      COMMON / BASE2 / ACC1,ACC2,ITMX1,ITMX2
      COMMON / BASE3 / SI,SI2,SWGT,SCHI,SCALLS,ATACC,NSU,IT,WGT
      INCLUDE  'incl1.f'
      COMMON /AMREG /MXREG
      COMMON /AMSPIN/JHS(NEXTRN), JHE(NEXTRN), ASPIN
      REAL*8     ANS0, ANS
*  P : Table of four momenta
* PP : Table of inner products
      REAL*8     XX(MXDIM),P(4,NEXTRN),PP(NEXTRN,NEXTRN)
      COMMON /SP4VEC/ VEC(4,NEXTRN)

*=====================================================================
*          Initialization
*=====================================================================
      ANSUM = 0.0D0
      DO  5 I = 1, NDIM
         XX(I) = X(I)
    5 CONTINUE

      NREG  = 1
      DFT   = 0.D0

*=====================================================================
*          Kinematics
*=====================================================================

      DO 1000 IREG = 1 , MXREG

         IF( IREG .GT. NREG ) GO TO 1000

         CALL KINEM(NEXTRN, XX, P, PP, YACOB,NREG,IREG,JUMP)
```

Source list 3.12 An example of `FUNC`

continue to the next page

```
*-----------------------------------------------------------------------
*          Reset the temporal buffer for the region 1
*-----------------------------------------------------------------------
        IF( IREG .EQ. 1 ) THEN
            DFT = 0.D0
            DO 180 K = 1, NEXTRN
            DO 180 J = 1, 4
               VEC(J,K) = 0.D0
  180       CONTINUE
        ENDIF
        IF( JUMP .NE. 0 ) GO TO 1000

*-----------------------------------------------------------------------
*          For user's cut
*-----------------------------------------------------------------------
C    CALL USRCUT(JUMP)
C    IF( JUMP .NE. 0 ) GOTO 1000
*-----------------------------------------------------------------------
*          Four momenta of external particles
*-----------------------------------------------------------------------
        DO 20 I = 1, 4
*                                           1:  EL-  INITIAL   LPRTCL
           PE0001(I) = P(I,  1)
*                                           2:  EL+  INITIAL   LANTIP
           PE0002(I) = P(I,  2)
*                                           3:  WB+  FINAL     LPRTCL
           PE0003(I) = P(I,  3)
*                                           4:  WB-  FINAL     LANTIP
           PE0004(I) = P(I,  4)
*                                           5:  AB   FINAL     LPRTCL
           PE0005(I) = P(I,  5)
   20   CONTINUE

*-----------------------------------------------------------------------
*          Inner products of momenta of external particles
*-----------------------------------------------------------------------
        DO 30 J = 1, NEXTRN
        DO 30 I = 1, NEXTRN
           PPROD(I, J) = PP(I, J)
   30   CONTINUE
*=======================================================================
*          Amplitude calculation
*=======================================================================
*       =============
         CALL AMPTBL
*       =============
```

Source list 3.12 An example of FUNC

continue to the next page

```
*        ==================
          CALL AMPSUM(ANSO)
*        ==================

         FKNORM = YACOB*ASPIN
         ANS    = ANSO*FKNORM
         ANSUM  = ANSUM + ANS

*------------------------------------------------------------------------
*          Save four momenta and probabilities of the region 1
*------------------------------------------------------------------------
         IF( IREG .EQ. 1 ) THEN
             DFT = ANS
             DO 420 K = 1, NEXTRN
             DO 420 J = 1, 4
                VEC(J,K) = P(J,K)
  420        CONTINUE
         ENDIF
*========================================================================
*          Fill Histograms and Scatter plots
*========================================================================
          DO 40 I = 1, NDIM
             CALL XHFILL( I, XX(I), ANS )
   40     CONTINUE

          K = 0
          DO 50 I = 1, NDIM-1
          DO 50 J = I+1, NDIM
             K    = K + 1
             CALL DHFILL( K, XX(I), XX(J), ANS )
   50     CONTINUE
*========================================================================
*          Update summary table
*========================================================================
          ANSP(0) = ANSP(0) + WGT*ANS
          DO 60 IGR = 1, JGRAPH
             ANSP(IGR)=ANSP(IGR) + WGT*YACOB*ASPIN*ANCP(IGR)
   60     CONTINUE
          NKCALL = NKCALL + 1
          IF( NKCALL .GT. 10000) THEN
              NKCALL = NKCALL - 10000
              FKCALL = FKCALL + 10000
          ENDIF

 1000 CONTINUE

      FUNC = ANSUM
```

Source list 3.12 An example of FUNC

continue to the next page

```
*-----------------------------------------------------------------------
*         Put the final 4 vectors into the arrays VEC()
*-----------------------------------------------------------------------
*     IF( FUNC .GT. 0.D0 ) THEN
*         IF( DFT/FUNC .LT. DRN(DUM)) THEN
*             DO 850 K = 1, NEXTRN
*             DO 850 J = 1, 4
*                VEC(J,K) = P(J,K)
* 850         CONTINUE
*         ENDIF
*     ENDIF

      RETURN
      END
```

Source list 3.12 An example of FUNC

3.5.5 Histogram package

The program package BASES/SPRING has its own histogram package, whose characteristics are as follows;

1) The buffer size of histograms and scatter plots is to be defined in the main program MAINBS.

   ```
   INCLUDE 'inclh.f'
   COMMON /PLOTB/ IBUF( 281*NHIST + 2527*NSCAT + 281 )
   ```

   The parameters NHIST and NSCAT are defined in the include file inclh.f. By changing these numbers one can make histograms and scatter plots up to 50 for each. The required buffer sizes for a histogram and a scatter plot are 281 and 2527 32-bit words, respectively.

2) Somewhere in USERIN, there should be a statement

   ```
   CALL BHINIT( NHIST, NSCAT )
   ```

   in order to let the system know the numbers of histograms and scatter plots.

3) To initialize the histograms and scatter plots, the following routines are to be called in USERIN.

```
      CALL XHINIT( ID#,
     .             lower_limit, upper_limit, # of bins, ' Title '),
```

```
and
  CALL DHINIT( ID#,
 .              x_lower_limit, x_upper_limit, # of x bins,
 .              y_lower_limit, y_upper_limit, # of y bins,
 .                                             ' Title '),
```

respectively.

4) To fill the histograms or the scatter plots on a scalar computer the following filling routines are called in the function FUNC:

```
  CALL XHFILL( ID#, V, FUNC )      for each histogram
  CALL DHFILL( ID#, VX, VY, FUNC)  for each scatter plot
```

5) The outputs of histograms and scatter plots can display even a *negative* function as well as the positive definite function.

6) The maximum number of bins both for histograms and scatter plots is 50.

3.5.6 Output from BASES

As described in section 3.5.1, there are several kinds of outputs from BASES, and we can select a combination of outputs by the print flag. The outputs consist of the following items.

1) **Job parameter**

   At the beginning of a job and just after reading the job parameters, their values are printed out as well as the number of nodes, which consist of the start and final loop counts, the print flag, the job input flag and the computing time limit as shown in the output 3.2.

2) **Parameters for BASES**

   After returning from USERIN, the parameters given there are printed out, some of which are numbers of the integration variables, the wild variable and the sampling points per iteration, N_{dim}, N_{wild} and $N_{call}^{(given)}$. From these numbers, the number of the small-regions per variable N_g, that of the sub-regions per variable N_s, that of real sampling points per iteration $N_{call}^{(real)}$, and that of hypercubes N_{cube} are calculated and printed. Further, for each integration variable, the lower and upper limits, XL(*i*) and XU(*i*), the grid optimization flag IG(*i*), and the kind of variable (*i.e.* wild or not) are printed. And finally the maximum iteration number and the expected accuracy both for the grid optimization and the integration steps are printed. An example of this output is given in the output 3.2.

```
                                                 Date: 93/ 1/ 9  14:02
   ***************************************************
   *                                                 *
   *    BBBBBB     AAA     SSSSS   EEEEEE   SSSSS    *
   *    BB   BB   AA AA   SS   SS  EE      SS   SS   *
   *    BB   BB  AA   AA  SS       EE      SS        *
   *    BBBBBB   AAAAAAA   SSSSS   EEEEEE   SSSSS    *
   *    BB   BB  AA   AA       SS  EE           SS   *
   *    BB   BB  AA   AA  SS   SS  EE      SS   SS   *
   *    BBBBBB   AA   AA   SSSSS   EEEEEE   SSSSS    *
   *                                                 *
   *                BASES Version 5.0                *
   ***************************************************


   <<  Parameters for this JOB  >>

      Current Loop Count =          1
      Maximum Loop Count =          1
      Print Flag         =          4
      JOB Input Flag     =          0
      Number of Nodes    =          1
      CPU Time Limit     = No limit


   <<   Parameters for BASES    >>

    (1) Dimensions of integration etc.
        # of dimensions :    Ndim  =        4   ( 50 at max. )
        # of Wilds      :    Nwild =        4   ( 15 at max. )
        # of sample points : Ncall =     4802 (real)     5000 (given)
        # of small regions : Ng    =       49 / variable
        # of subregions    : Ns    =        7 / variable
        # of Hypercubes    : Ncube =     2401

    (2) About the integration variables
        ------+---------------+---------------+-------+-------
            i       XL(i)           XU(i)       IG(i)   Wild
        ------+---------------+---------------+-------+-------
            1    0.000000E+00    1.000000E+00     1      yes
            2    0.000000E+00    1.000000E+00     1      yes
            3    0.000000E+00    1.000000E+00     1      yes
            4    0.000000E+00    1.000000E+00     1      yes
        ------+---------------+---------------+-------+-------

    (3) Parameters for the grid optimization step
        Max.# of iterations: ITMX1 =        5
        Expected accuracy  : Acc1  =    .2000 %

    (4) Parameters for the integration step
        Max.# of iterations: ITMX2 =        5
        Expected accuracy  : Acc2  =    .0100 %
```

Output 3.2 General information of the integration

3) **Convergency behavior**

According to the print flag the two kinds of convergency behaviors can be obtained, one is for the grid optimization step and another is for the integration step. The print format consists of the result of each iteration and the cumulative result and the computing time used.

In the result of each iteration, the sampling efficiency (the percentage of the points inside of the kinematical boundary), the ratio of the numbers of the negative valued sampling points to the total number of sampling points in unit of percent, the estimate of integral of the iteration and the estimated accuracy in unit of percent are shown.

In the cumulative result, the cumulative estimates of integral and error are listed up in addition to the accuracy in the unit of percent. The computing time in this table is measured from the beginning of the grid optimization step till the end of the current iteration, which does not contain the time of overhead but that used for estimating integral.

In the convergency behavior for the grid optimization step, it should be checked that the accuracy for each iteration does decrease iteration by iteration and converge to a stable value. If not the case, it is recommended to increase the number of sampling points N_{call} and try again. When the increment of number of sampling points does not help to improve the behavior, the current choice of the integration variables may not be suitable for the behavior of integrand. Examples of convergency behavior both for the grid optimization and integration steps are given in the outputs 3.3 and 3.4, respectively.

```
                                                 Date: 93/ 1/ 9  14:02
            Convergency Behavior for the Grid Optimization Step
--------------------------------------------------------------------------
<- Result of  each iteration -> <-     Cumulative Result     -> < CPU  time >
 IT Eff R_Neg   Estimate  Acc % Estimate(+- Error )order  Acc % ( H: M: Sec )
--------------------------------------------------------------------------
  1  94   .00  3.298E+00  3.406  3.298227(+- .112323)E 00  3.406   0: 1: 9.79
  2  96   .00  3.500E+00  1.670  3.457213(+- .051841)E 00  1.500   0: 2:20.89
  3  97   .00  3.417E+00  1.045  3.429967(+- .029414)E 00   .858   0: 3:32.87
  4  97   .00  3.400E+00  1.031  3.417499(+- .022535)E 00   .659   0: 4:45.09
  5  97   .00  3.384E+00   .911  3.405825(+- .018189)E 00   .534   0: 5:57.28
--------------------------------------------------------------------------
```

Output 3.3 Convergency behavior for the grid optimization step

```
                                                   Date: 93/ 1/ 9  14:02
                Convergency Behavior for the Integration Step
------------------------------------------------------------------------------
<- Result of  each iteration ->  <-     Cumulative Result     -> < CPU  time >
 IT Eff R_Neg   Estimate  Acc %  Estimate(+- Error )order  Acc % ( H: M: Sec )
------------------------------------------------------------------------------
  1  97   .00  3.449E+00   .987  3.449476(+- .034036)E 00   .987   0: 7: 9.38
  2  97   .00  3.370E+00   .944  3.407093(+- .023241)E 00   .682   0: 8:22.18
  3  97   .00  3.391E+00   .956  3.401467(+- .018889)E 00   .555   0: 9:33.93
  4  97   .00  3.378E+00   .966  3.395686(+- .016350)E 00   .481   0:10:45.54
  5  97   .00  3.383E+00   .932  3.392969(+- .014514)E 00   .428   0:11:58.49
------------------------------------------------------------------------------
```

Output 3.4 Convergency behavior for the integration step

The accuracy of each iteration must be stable in the integration step. When the integration variables does not suit for the integrand, it fluctuates iteration by iteration and may jump suddenly to a big value in the worst case.

In the interactive mode the convergency behavior is printed iteration by iteration, while it is printed only for the final 50 iterations in the batch mode. This mode is to be selected at the installation time by setting the flag "`INTV`" in the routine `BSMAIN`.

4) **Histograms and scatter plots**

If histograms and scatter plots are initialized in `USERIN` and filled in `FUNC`, their results are printed at each end of the grid optimization step and the integration step according to the print flag. In the output 3.5 we show only the histogram `ID` = 3 for saving the space of this manual.

The first and the last bins of histogram represent values of the underflow and the overflow bins, respectively. The first column shows the lower edge value of each histogram bin. The second column represents the estimated differential value and error after the characters "`+-`", both of which are to be multiplied by a factor "`E xx`" shown as order. On the right hand side of these columns a histogram of the differential values is drawn both in the linear scale with "`*`" and in the logarithmic scale with "`O`". If negative values exist in some bins only the linear scale histogram is shown.

The scatter plot represents only the relative height of the function. The height of the function value is described by ten characters; `1`, `2`, `3`, ..., `8`, `9` and `*`, while the depth (for the negative values) is displayed by ten characters; `a`, `b`, `c`, `d`, ..., `h`, `i` and `#`. The point which has a negative value but larger than the value of the level "`a`" is indicated by "`-`". On the other hand, the point describing a positive value but less than the level "`1`" is given either by "+" (if a negative value exists somewhere) or by "`.`" (if only the positive values exist). In the output 3.6 an example of scatter plot is shown.

```
Histogram (ID =  3 ) for X( 3) SPECTRUM
                             Linear Scale indicated by "*"
    x      d(Sigma)/dx     0.0E+00      8.3E+00      1.7E+01     2.5E+01
+-------+------------------+------------+------------+-----------+-----------+
I  E  0 I  .000        E  0I                                                 I
I  .000 I 2.595+- .051 E  1I****************************************000000000I
I  .020 I 8.402+- .279 E  0I*************00000000000000000000000000          I
I  .040 I 5.954+- .265 E  0I**********00000000000000000000000000             I
I  .060 I 4.817+- .229 E  0I********00000000000000000000000000               I
I  .080 I 3.415+- .171 E  0I******0000000000000000000000000                  I
I  .100 I 3.698+- .167 E  0I******00000000000000000000000000                 I
I  .120 I 3.053+- .148 E  0I*****0000000000000000000000000                   I
I  .140 I 2.898+- .154 E  0I*****0000000000000000000000000                   I
I  .160 I 2.443+- .160 E  0I****000000000000000000000000                     I
I  .180 I 2.917+- .155 E  0I*****0000000000000000000000000                   I
I  .200 I 2.183+- .134 E  0I****00000000000000000000000                      I
I  .220 I 1.874+- .127 E  0I***00000000000000000000000                       I
I  .240 I 2.161+- .129 E  0I****00000000000000000000000                      I
I  .260 I 1.924+- .123 E  0I***00000000000000000000000                       I
I  .280 I 1.589+- .110 E  0I***0000000000000000000000                        I
I  .300 I 1.543+- .100 E  0I***000000000000000000000                         I
I  .320 I 1.387+- .107 E  0I***00000000000000000000                          I
I  .340 I 1.316+- .086 E  0I**000000000000000000000                          I
I  .360 I 1.239+- .091 E  0I**00000000000000000000                           I
I  .380 I 1.282+- .091 E  0I**000000000000000000000                          I
I  .400 I 1.021+- .069 E  0I**0000000000000000000                            I
I  .420 I 9.757+- .814 E -1I**000000000000000000                             I
I  .440 I 9.046+- .458 E -1I**000000000000000000                             I
I  .460 I 8.947+- .466 E -1I**000000000000000000                             I
I  .480 I 9.287+- .679 E -1I**000000000000000000                             I
I  .500 I 8.258+- .790 E -1I**00000000000000000                              I
I  .520 I 9.083+- .589 E -1I**000000000000000000                             I
I  .540 I 9.549+- .668 E -1I**000000000000000000                             I
I  .560 I 9.718+- .739 E -1I**000000000000000000                             I
I  .580 I 9.076+- .840 E -1I**000000000000000000                             I
I  .600 I 1.193+- .088 E  0I**00000000000000000000                           I
I  .620 I 1.401+- .093 E  0I***00000000000000000000                          I
I  .640 I 1.428+- .119 E  0I***000000000000000000000                         I
I  .660 I 1.097+- .101 E  0I**0000000000000000000                            I
I  .680 I 1.947+- .123 E  0I***00000000000000000000000                       I
I  .700 I 1.979+- .126 E  0I***00000000000000000000000                       I
I  .720 I 2.099+- .122 E  0I****00000000000000000000000                      I
I  .740 I 2.214+- .138 E  0I****00000000000000000000000                      I
I  .760 I 1.784+- .125 E  0I***00000000000000000000000                       I
I  .780 I 2.376+- .141 E  0I****000000000000000000000000                     I
I  .800 I 2.483+- .144 E  0I****000000000000000000000000                     I
I  .820 I 2.566+- .149 E  0I****0000000000000000000000000                    I
I  .840 I 2.883+- .152 E  0I*****0000000000000000000000000                   I
I  .860 I 2.803+- .150 E  0I*****000000000000000000000000                    I
I  .880 I 3.448+- .175 E  0I******0000000000000000000000000                  I
I  .900 I 3.906+- .176 E  0I******00000000000000000000000000                 I
I  .920 I 4.146+- .196 E  0I*******00000000000000000000000000                I
I  .940 I 5.760+- .247 E  0I*********000000000000000000000000000             I
I  .960 I 8.741+- .291 E  0I**************0000000000000000000000000          I
I  .980 I 2.612+- .052 E  1I****************************************000000000I
I  E  0 I  .000        E  0I                                                 I
+-------+------------------+-------------------+-------------------+---------+
    x      d(Sigma)/dx     1.0E-01             1.0E+00             1.0E+01
                             Logarithmic Scale indicated by "0"
```

Output 3.5 An example of histogram

```
Scat_Plot (ID =  6 ) for X( 3)-X( 4) DISTRIBUTION

  E  0    +--------------------------------------------------+
  .980    I8212.1.............. .. . ........1.1.....1..1137I
  .960    I8332..1111.1.......... ...................11.114*I
  .940    I832.................. .  .. ............1.1.212*I
  .920    I821.11.1................................1.1.13*I
  .900    I722111.1.....1................. ........11.11236I
  .880    I821..1.............. ..................1.....1118I
  .860    I7111.........1.......... .............1..1.1.1236I
  .840    I821.11.1.1..1........... ...... .1....1.....11227I
  .820    I831.11.1.....1......... ...... ....1.....1.1...117I
  .800    I821211....11..... .......1 ....1.......111...1.227I
  .780    I421.1.1..1...... ..............1. .....11.111.128I
  .760    I711....111..1..........1..1..........1....1111124I
  .740    I421.1.1..1............ ..........1........111116I
  .720    I5212.1...11............ ..... ......1..........16I
  .700    I811..211............. .. ....... ...1.......11117I
  .680    I5211.11...........................1.....1.....26I
  .660    I8111..........1....... ... .....1......1..1.1..137I
  .640    I41...1..1.................... .....1...1..1111216I
  .620    I72.1.1........... .........................11127I
  .600    I7221.1.1...1....... ...... .. .. ...........1.1137I
  .580    I7211.1.........1....... .. .........1.....1.111.15I
  .560    I52...11.1........ ......................1....1.136I
  .540    I52111........11..... ...... ..........1...1....116I
  .520    I5112.....1......1.. ...... ..... ............1224I
  .500 Y  I5221....1.....1.. ......................11.1.1138I
  .480    I72...1............... . . ....... .......1....1125I
  .460    I5.21.11.....1...1.....................1.......126I
  .440    I611......1....... ..................11........2.17I
  .420    I7111.11.11.............................1......126I
  .400    I622...... .....1........   . ................1125I
  .380    I73..1.11........... . . .... .........1....111114I
  .360    I51111....1.......1.... . .........1......1..1.1.18I
  .340    I8211..1... 1........... ..........1........21118I
  .320    I4211.....1......... .. ...... ........1.1111..26I
  .300    I52111.11.......1........ . . .......1... ....1.227I
  .280    I611..1...1...11..1........ .... ...1...1..1.111127I
  .260    I8111..................................... 1...1.215I
  .240    I62.1.1....1.... ..... ... . .............1..1.234I
  .220    I9111.12............. ....................1..11115I
  .200    I63.111..1......... ... ...........11....1..1...128I
  .180    I6211..1..1..................  ...........1.1.1128I
  .160    I512..1......1........1.........  .....1.11...1.127I
  .140    I8221.....1......1........1.... .........1..1.21217I
  .120    I83111..1.1..1...........1...  .1........1121.11139I
  .100    I7122..11..... ........... ... . .....1...1...12127I
  .080    I821.1....1............   .....................237I
  .060    I72121111.....1.......... ............1......11147I
  .040    I93311111.1.............. ... .....1..1...1.11.11*I
  .020    I9211..1.......1.... .....................1.12.1238I
  .000    I8232...1.1..1......................1...1...1..327I
  E  0    +--------------------------------------------------+
 Low-                             X
 Edge
            00000000000000000000000000000000000000000000000000
  E  0      ..................................................
            00000011111222223333344445555556666777777888899999
 Low-       01367923579245791457914580135791358014579235801367
 Edge       099099099990099990999909009999999990090999099009909
```

Output 3.6 An example of scatter plot

5) **Message at the JOB termination**

At the end of the job a message from BASES is printed. When the job was terminated due to the shortage of the computing time, the following message is given:

```
**** Computing time out ****
   Next Loop Count =    1
   Next JOB Flag   =    3
```

Submitting a successive job with the loop count 1 and the job flag 3, the integration can be continued further. If the job is terminated normally 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
```

It may happen that the accuracy of the integration is not small enough even after the normal termination of job. Because a job is terminated not only by the accuracy, but also by the iteration number. In this case we can continue the integration by giving the maximum iteration number larger than the previous job and by setting the job flag 3.

```
******   Computing Time Information   ******

Start at: 93/ 1/ 9  14:02
End   at: 93/ 1/ 9  14:26

(1) For this JOB          H: M:  Sec
    Overhead            :   0: 0: 0.18
    Grid Optim. Step    :   0: 5:57.28
    Integration Step    :   0: 6: 1.20
    JOB elapsed time    :   0:11:58.67

(2) For Total calculation  H: M:  Sec
    Overhead            :   0: 0: 0.18
    Grid Optim. Step    :   0: 5:57.28
    Integration Step    :   0: 6: 1.20

(3) Expected event generation time
    Expected time for 1000 events :     21.49 Sec

******   Computing Time Information   ******
```

Output 3.7 Computing time information

6) **List of computing time**

As well as the message, a list of computing time is printed at the end of the job as shown in a output 3.7.

When the integration has been achieved by a single job, the items (1) and (2) are exactly the same. If the integration is performed by several jobs, the computing time is only for the current job, while that for total calculation includes all computing time from the beginning. The expected event generation time is printed at the item (3). From this value, the computing time limit for the event generation will be evaluated.

7) **Final result of integration**

When the print flag is set equal to "1" or "2", the final result of the integration step is printed, which consists of the loop number, the cumulative estimate and error, the number of iterations both for the grid optimization and the integration steps and the computing time used.

```
------------------------------------------------------
 Loop#  Estimate(+- Error )order  It1  It2 ( H: M: Sec )
------------------------------------------------------
     1  3.392969(+-0.014514)E 00    5    5   0:11:58.49
------------------------------------------------------
```

Output 3.8 Result of integration

When we calculate the cross section at several energy points by using the loop option, we can obtain the values of the cross section in the form of a table if the print flag is set equal to "1". It is noted that this computing time is reset at the beginning of each loop count.

8) **Probability information**

Before terminating the integration job, BASES generates a data file by the routine BSWRIT, where

1) Probability information
Probability of each hypercube, according to which a hypercube is sampled in the event generation.

2) The maximum values of integrand
The maximum value of integrand in each hypercube is stored, by which the sampling point are tested by using a uniform random number.

3) Contents of histograms
In the event generation, those histograms are printed out comparing to the distribution of generated events which are defined in the integration by

`BASES`. For this purpose, the contents of histograms taken in the `BASES` are stored in this file.

4) Control data for `BASES`
   By giving the job flag non zero value, the integration can be continued further as described in section 3.5.1. For this purpose, the control data as well as the results up to the current job are stored in this file.

Although there are several versions of `BASES/SPRING`, *e.g.* the original `BASES /SPRING`, `BASES25/SPRING25`, and `BASES50/SPRING50`, the data format of this file *does* depend on the version. The newest one is `BASES50/SPRING50` and is recommended to use. We call `BASES50/SPRING50` as `BASES/SPRING` throughout this manual. *Be careful not to use the different versions for* `BASES` *and* `SPRING`.

3.6 Event generation

As described in section 2.8, an advantage of BASES/SPRING packages is that if a differential cross section is integrated by BASES the four vectors of final state particles are easily generated with weight one by using SPRING. In this section, a description of SPRING is given in the following order.

(1) Input for SPRING

(2) Program structure of SPRING

(3) Specifications of the subprograms to be prepared

(4) Output from SPRING

The event generation by SPRING is normally quite fast. But if calculation of the integrand requires much computing time, both the integration and the event generation takes much time. For such a case we recommend to use a vector computer if available. A vector version of SPRING will be described in section 5.

3.6.1 Input for SPRING

There are two inputs for SPRING. One is a file of the probability information for each hypercube, which is produced by the integration package BASES. In this file the following data are saved:

(a) The probability of sampling each hypercube.

(b) The maximum value of integrand in each hypercube.

(c) The contents of histograms and scatter plots.

(d) The control data for BASES.

SPRING with a different version from that of BASES should not be used for the event generation, since the data format of this file *does* depend on the version as mentioned in the previous section. The most new one is BASES50/SPRING50 and is recommended to use.

Another input is something like the job parameter. At the beginning of the generation job, the following parameters are read from the logical unit 5:

(1) The number of events to be generated.

(2) The computing time limit in unit of minutes.
Even if an UNIX system is used, this parameter should be given to avoid an infinite loop as described in subsection 3.6.2.

The event generation loop is terminated not only by the generation of given numbers of events, but also by lack of the remaining computing time.

Fig. 3.5 Program structure of `SPRING`

3.6.2 Program structure of SPRING

In figure 3.5, the program structure of SPRING is shown, where the subprograms in the white box are generated by GRACE automatically and are to be finalized by user. Others are included in the BASES/SPRING library or CHANEL library.

In the source list 3.13, the main program MAINSP generated by GRACE is shown. At the beginning of the main, the common /PLOTB/ is declared to keep enough buffer for the histograms similar to MAINBS for the integration. Then it calls the steering routine SPMAIN, whose arguments are the entry name of the integrand function and a parameter MXTRY. The parameter MXTRY defines the maximum number of trials for getting an accepted event, which make the event generation free from an infinite loop described later in this subsection. When the four vectors of generated events are going to be written on a file, the file must be opened here.

```
* Main program for SPRING
      IMPLICIT REAL*8(A-H,O-Z)

      INCLUDE 'inclh.f'
      COMMON /PLOTB/ IBUF( 281*NHIST + 2527*NSCAT + 281 )

      EXTERNAL FUNC

      WRITE(*,'(10X,A//)')'* 5120     E+ E-  => W+  W-  A       TREE '

      open(23,file='bases.data',status='old',form='unformatted')

      MXTRY = 50

      CALL SPMAIN(FUNC, MXTRY)

      close(23)

      STOP
      END
```

Source list 3.13 The main program MAINSP

The program flow in SPMAIN is as follows;

(A) Initialization

(1) At the beginning, the number of generated events and computing time limit are read.

(2) By the subroutine BSREAD the probability information of all hypercubes and the contents of histograms and scatter plots are read from a binary file.

(3) USERIN is called for initialization of histograms *etc.* and kinematics.

(4) The probability distribution read from the file is changed into the cumulative distribution.

(5) The subprogram `SPINIT` is called where we can initialize the additional histograms and scatter plots if we want. The description of additional histograms and scatter plots is given later in this section.

If user wants to use other histogram package (*e.g.* Handypack or HBOOK) for taking the event distribution, the initialization of these package should be done in `SPINIT`.

(B) Event generation loop

(1) A hypercube (say the i-th hypercube) is sampled according to its probability by a random number generated by a function `DRN`.

(2) A point is sampled in a small region in the i-th hypercube, sampled in the step (B.1).

(3) The value of the integrand at the sampled point ζ is calculated by calling `FUNC`.

(4) If the sampled point ζ satisfies the condition

$$\frac{f(\zeta)}{p(\zeta)}/Max.[\frac{f(x_i)}{p(x_i)}] < \eta \text{ (= a uniform random number)},$$

then this point is accepted as an event, and go out of the event generation loop.

(5) If the sampled point is not accepted and the number of trials to get an event is less than the given value of `MXTRY`, the histogram information for the point is cleared by the subroutine `SHCLER` and go to the step (B.2).

(6) If the number of trials is larger than the given value, this hypercube is abandoned, and go to the step (B.1).

(C) Four vectors of generated events
When a point is accepted as an event, the subprogram `SPEVNT` is called, where the four vectors of final state particles of the accepted event are calculated and written onto a output file. If the additional histograms and scatter plots are defined in `SPINIT`, they are filled in this routine. If other histogram package is used, their filling routines are called here. This routine should be coded by user oneself.

(D) Check the number of events
Increment the number of generated event and test the remaining computing time. If the number of events is less than the given number or there remains enough computing time for generating one event, go to the step (B.1).

(E) Termination
Before terminating the job, histograms and scatter plots are printed by SHPLOT. If other histogram package is used, some print routines should be called in the subprogram SPTERM.

As described in the step (B.5), the parameter MXTRY plays an important role. Without limiting the maximum number of trials to get an event, the generation loop may come into an infinite loop. This parameter is set in the main program MAINSP and default number is equal to 50.

3.6.3 Subprograms to be prepared

To use BASES the main program MAINBS and the subprograms USERIN, KINIT, FUNC, KINEM and USROUT are to be prepared by user. They are generated by GRACE automatically and are left for user to finalize. These subprograms are also necessary for the event generation by SPRING except for MAINBS and USROUT. As their specifications can be found in subsections 3.5.2, 3.5.3 and 3.5.4. we will not repeat them here unless there exist difference between their specifications in BASES and SPRING. In addition to them, SPRING requires the main program MAINSP and subprograms SPINIT, SPEVNT and SPTERM.

No change

Main program MAINSP is produced by GRACE in a complete form. Subprograms USERIN, KINIT and KINEM, used in BASES, does not need to be modified. Especially the subprogram USERIN should be identical to that used in BASES.

FUNC

When the integrand is a single-valued function, it *should not be changed.* But if it is a two-valued function, the last part of the function code must be activate, which part is normally commented out. The example of this case is shown in the source list 3.14.

When the kinematics is described by a many-valued function, a sample point in the integration volume corresponds to several distinct points in the phase space, for each of which differential cross section is calculated. In the integration the values of differential cross section at these points are simply summed and the sum is given as the function value FUNC, while in the event generation a point among these points must be sampled according to their probabilities.

The example in the source list 3.14 and 3.12 shows the two-valued function case. For the first point the four vectors and numerical value of the differential cross section are stored in an arrays VEC(j, k) and variable DFT at the do loop 420 in the list 3.12. If the ratio of DFT and FUNC is less than a random number, the second point in the phase space is taken as a sampled point, where FUNC is the sum of the differential cross section values at these two points. This method can be easily extended to a many-valued function case.

```
 1000 CONTINUE

      FUNC = ANSUM

*-------------------------------------------------------------
*       Put the final 4 vectors into the arrays VEC()
*-------------------------------------------------------------
      IF( FUNC .GT. 0.D0 ) THEN
          IF( DFT/FUNC .LT. DRN(DUM)) THEN
              DO 850 K = 1, NEXTRN
              DO 850 J = 1, 4
                 VEC(J,K) = P(J,K)
  850         CONTINUE
          ENDIF
      ENDIF

      RETURN
      END
```

Source list 3.14 The last part of `FUNC`

SPINIT

Subprogram `SPINIT`, generated by `GRACE`, is just dummy. Before going into the generation loop, this subprogram is called for initialization.

If you want to make some histograms or scatter plots in addition to those introduced in `BASES`, each histogram or scatter plot should be initialized here. In this meaning these histograms or scatter plots defined here are called as the additional histograms or scatter plots. An example is shown in the source list 3.15.

```
      SUBROUTINE SPINIT

      CALL XHINIT( 5, 0.D0, 180.D0, 36,'Photon Angular dist.')

      RETURN
      END
```

Source list 3.15 An example of `SPINIT`

If someone wants to use other histogram package like Handypack or HBOOK, initialization of these programs should be done in this routine.

SPEVNT

Subprogram `SPEVNT`, generated by `GRACE`, is a dummy routine. Only when a event is accepted, this routine is called. The four vectors of generated event are to be written on a file here. The function program `FUNC`, generated by `GRACE`, has the common variable VEC(4,NEXTRN), where the four vectors of initial and final particles are stored.

If the additional histograms or scatter plots are defined in SPINIT, the filling routines XHFILL or DHFILL are to be called here. Am example to take the polar angle distribution of the photon is in the source list 3.16. For other histogram packages, the filling routines should be called here.

SPTERM

Subprogram SPTERM, generated by GRACE, is dummy. This routine is called just before terminating the process. When other histogram package is used, the output routine should be called in this routine.

It should be noted that the subprograms USERIN and FUNC must not be changed. If some part is changed and it gives a different behavior of the integrand, the generation efficiency of SPRING might become very low and the completely wrong event samples might be generated because the resultant differential cross section does not match to the probability distribution in the input file of SPRING. The same situation can emerge when events are generated by using the differential cross section of process A with the input file produced by using a different process B.

```
      SUBROUTINE SPEVNT
      IMPLICIT REAL*8 ( A-H, O-Z)
*
      PARAMETER (NGRAPH =   28, NEXTRN =   5, LAG =  72)
      COMMON /SP4VEC/ VEC(4,NEXTRN)
      COMMON / AMCNST / PI, PI2, RAD, GEVPB, ALPHA
*
*     Calculate photon angular dist.

      PP    = SQRT((VEC(1,1)**2+VEC(2,1)**2+VEC(3,1)**2)
     .            *(VEC(1,5)**2+VEC(2,5)**2+VEC(3,5)**2))

      CS    =( VEC(1,1)*VEC(1,5)+VEC(2,1)*VEC(2,5)
     .        +VEC(3,1)*VEC(3,5) )/PP
      TH    = ACOS(CS)*180.D0/PI
      IF( TH .LT. 0.D0 ) THEN
          PRINT *,TH,CS
      ENDIF

      CALL XHFILL( 5, TH, PP )

      RETURN
      END
```

Source list 3.16 An example of SPEVNT

3.6.4 Output from SPRING

The output from SPRING consists of the general information, histogram output, the number of trials distribution and the four vector output. There are two kinds of histogram output, one is the original histogram and other is the additional histogram.

General information

After generating events, the following information is printed:

```
                                                  Date: 93/ 1/20  22:20
****************************************************
*                                                  *
*    SSSS   PPPPP   RRRRR   IIII  N   NN   GGGG    *
*   SS  SS  PP  PP  RR  RR   II   NN  NN  GG  GG    *
*   SS      PP  PP  RR  RR   II   NNN NN  GG        *
*    SSSS   PPPPP   RRRR     II   NNN NN  GG GGGG  *
*       SS  PP      RR RR    II   NN NNN  GG  GG    *
*   SS  SS  PP      RR  RR   II   NN  NN  GG  GG    *
*    SSSS   PP      RR   RR IIII  NN   N   GGGG    *
*                                                  *
*                SPRING Version 5.0                *
****************************************************

Number of generated events    =     10000
Computing time for generation =   270.260 Seconds
               for Overhead   =      .230 Seconds
               for Histograms =      .650 Seconds
GO time                       =   271.140 Seconds
Max. number of trials MXTRY =        50 per event
Number of mis-generations   =         0
```

Output 3.9 General information of the event generation

When the number of trials to generate one event exceeds the number MXTRY, this outbreak is counted as the number of mis-generation. If this number is not negligible small, something happens in the event generation, *e.g.* mis-match between the integrand and the probability information of the input file, or the grids determined by BASES are not enough optimized. This can be also checked by the number of trials distribution described later.

Histograms

There are two kinds of histograms.

One is the original histogram, which is defined in the integration stage by BASES. The contents of these histograms produced in the integration are read from the input file and are compared with the frequency distribution taken in the event generation. This comparison is done in the logarithmic scale, where the statistical error of each bin is represented by "< >". If error is smaller than the two character space, only the frequency is shown by "O". The histogram obtained by BASES is represented by "*". An example of the original histogram is shown in the output

3.10, which can be compared with the histogram shown in the output 3.5 of section 3.5.6.

Another kind of histogram is the additional histogram, which is defined in `SPINIT`. Since there is no data taken in the integration stage for the additional histogram, only the frequency distribution is displayed, whose example is given in the output 3.11.

```
Original Histogram (ID =  3 ) for X( 3) SPECTRUM
Total =     10000 events   "*" : Orig. Dist. in Log Scale.
    x      d(Sig/dx)  dN/dx 1.0E-01             1.0E+00             1.0E+01
+--------+----------+-------+-------------------+-------------------+---------+
I   E  0 I  .000E  0I      0I                                                 I
I   .000 I 2.595E  1I   1530I************************************************0I
I   .020 I 8.402E  0I    502I**************************************0          I
I   .040 I 5.954E  0I    358I***********************************<>            I
I   .060 I 4.817E  0I    265I********************************<>               I
I   .080 I 3.415E  0I    217I******************************<>                 I
I   .100 I 3.698E  0I    207I******************************<>                 I
I   .120 I 3.053E  0I    181I*****************************<>                  I
I   .140 I 2.898E  0I    171I****************************<>                   I
I   .160 I 2.443E  0I    178I****************************<0>                  I
I   .180 I 2.917E  0I    139I**************************<0>*                   I
I   .200 I 2.183E  0I    119I*************************<>                      I
I   .220 I 1.874E  0I    118I*************************<>                      I
I   .240 I 2.161E  0I    112I************************<0>                      I
I   .260 I 1.924E  0I    109I************************<0>                      I
I   .280 I 1.589E  0I    117I*************************<>                      I
I   .300 I 1.543E  0I    109I************************<0>                      I
I   .320 I 1.387E  0I     90I**********************<0>                        I
I   .340 I 1.316E  0I     86I**********************<0>                        I
I   .360 I 1.239E  0I     80I*********************<0>                         I
I   .380 I 1.282E  0I     56I******************<0>**                          I
I   .400 I 1.021E  0I     49I*****************<0>*                            I
I   .420 I 9.757E -1I     71I********************<0>                          I
I   .440 I 9.046E -1I     49I*****************<0>                             I
I   .460 I 8.947E -1I     49I*****************<0>                             I
I   .480 I 9.287E -1I     44I****************<0>*                             I
I   .500 I 8.258E -1I     48I****************<*0>                             I
I   .520 I 9.083E -1I     31I************<*0>****                             I
I   .540 I 9.549E -1I     45I****************<0>*                             I
I   .560 I 9.718E -1I     49I*****************<0>                             I
I   .580 I 9.076E -1I     56I******************<0>                            I
I   .600 I 1.193E  0I     68I********************<0>                          I
I   .620 I 1.401E  0I     73I********************<0>                          I
I   .640 I 1.428E  0I     82I*********************<0>                         I
I   .660 I 1.097E  0I     95I*********************  <>                        I
I   .680 I 1.947E  0I     96I***********************<0>                       I
I   .700 I 1.979E  0I    131I**************************<>                     I
I   .720 I 2.099E  0I    135I**************************<>                     I
I   .740 I 2.214E  0I    110I************************<0>                      I
I   .760 I 1.784E  0I    128I*************************<0>                     I
I   .780 I 2.376E  0I    119I*************************<>*                     I
I   .800 I 2.483E  0I    117I*************************<>*                     I
I   .820 I 2.566E  0I    169I****************************<>                   I
I   .840 I 2.883E  0I    162I****************************<>                   I
I   .860 I 2.803E  0I    176I****************************<0>                  I
I   .880 I 3.448E  0I    215I******************************<>                 I
I   .900 I 3.906E  0I    206I******************************<>                 I
I   .920 I 4.146E  0I    260I********************************<>               I
I   .940 I 5.760E  0I    325I**********************************<>             I
I   .960 I 8.741E  0I    575I***************************************<>        I
I   .980 I 2.612E  1I   1523I************************************************0I
I   E  0 I  .000E  0I      0I                                                 I
+--------+----------+-------+-------------------+-------------------+---------+
    x      d(Sig/dx)  dN/dx    "0" : Generated Events.( Arbitrary unit in Log )
```

Output 3.10 An example of the original histogram

```
Additional Histogram (ID =  5 ) for Photon Angular dist.
Total =      9999 events   "*" : No. of events in Linear scale.
    x      Lg(dN/dx)  dN/dx 0.0E+00      1.4E+03     2.8E+03     4.1E+03
+--------+----------+-------+------------+------------+-----------+-----------+
I   E  2 I  .000E  0I      0I                                                 I
I   .000 I 4.386E  3I   4386I****************************************000000   I
I   .050 I 2.430E  2I    243I***00000000000000000000000000                    I
I   .100 I 8.800E  1I     88I*00000000000000000000000                         I
I   .150 I 6.900E  1I     69I*000000000000000000000                           I
I   .200 I 3.900E  1I     39I*000000000000000000                              I
I   .250 I 3.200E  1I     32I*00000000000000000                               I
I   .300 I 1.500E  1I     15I*0000000000000                                   I
I   .350 I 2.200E  1I     22I*000000000000000                                 I
I   .400 I 1.300E  1I     13I*000000000000                                    I
I   .450 I 6.000E  0I      6I*00000000                                        I
I   .500 I 9.000E  0I      9I*0000000000                                      I
I   .550 I 6.000E  0I      6I*00000000                                        I
I   .600 I 1.100E  1I     11I*000000000000                                    I
I   .650 I 1.100E  1I     11I*000000000000                                    I
I   .700 I 1.300E  1I     13I*000000000000                                    I
I   .750 I 4.000E  0I      4I*000000                                          I
I   .800 I 6.000E  0I      6I*00000000                                        I
I   .850 I 8.000E  0I      8I*0000000000                                      I
I   .900 I 6.000E  0I      6I*00000000                                        I
I   .950 I 9.000E  0I      9I*0000000000                                      I
I  1.000 I 7.000E  0I      7I*000000000                                       I
I  1.050 I 8.000E  0I      8I*0000000000                                      I
I  1.100 I 8.000E  0I      8I*0000000000                                      I
I  1.150 I 1.300E  1I     13I*000000000000                                    I
I  1.200 I 7.000E  0I      7I*000000000                                       I
I  1.250 I 1.200E  1I     12I*000000000000                                    I
I  1.300 I 9.000E  0I      9I*0000000000                                      I
I  1.350 I 1.100E  1I     11I*000000000000                                    I
I  1.400 I 2.200E  1I     22I*000000000000000                                 I
I  1.450 I 3.200E  1I     32I*00000000000000000                               I
I  1.500 I 2.600E  1I     26I*0000000000000000                                I
I  1.550 I 3.300E  1I     33I*00000000000000000                               I
I  1.600 I 5.200E  1I     52I*00000000000000000000                            I
I  1.650 I 9.100E  1I     91I*00000000000000000000000                         I
I  1.700 I 1.750E  2I    175I**00000000000000000000000000                     I
I  1.750 I 4.497E  3I   4497I*****************************************00000   I
I   E  2 I  .000E  0I      0I                                                 I
+--------+----------+-------+------------+------------+-----------+-----------+
    x      Lg(dN/dx)  dN/dx 1.0E+00      1.0E+01    1.0E+02      1.0E+03
                             "0" : No. of Events in Log. scale.
```

Output 3.11 An example of the additional histogram

Number of trials distribution

The number of trials distribution is printed out at the final stage, by which we can see how efficient the event generation was. The first column represents the number of trials to obtain one event and the number of events is shown in the third column. An example for the process $e^+e^- \to W^+W^-\gamma$ is shown in the output 3.12, where about 80% of events are generated with the first trial.

```
    ************* Number of trials to get an event *************
Total =     10000 events   "*" : No. of events in Linear scale.
    x      Lg(dN/dx)  dN/dx 0.0E+00      2.3E+03      4.5E+03     6.8E+03
+--------+----------+-------+------------+------------+-----------+-----------+
I   E  1 I  .000E  0I      0I                                                 I
I   .100 I 6.777E  3I   6777I**************************************0000000000 I
I   .200 I 1.826E  3I   1826I***********000000000000000000000000000000        I
I   .300 I 6.110E  2I    611I****0000000000000000000000000000000              I
I   .400 I 2.770E  2I    277I**00000000000000000000000000000                  I
I   .500 I 1.560E  2I    156I*000000000000000000000000000                     I
I   .600 I 1.080E  2I    108I*0000000000000000000000000                       I
I   .700 I 6.000E  1I     60I*0000000000000000000000                          I
I   .800 I 3.600E  1I     36I*0000000000000000000                             I
I   .900 I 3.400E  1I     34I*0000000000000000000                             I
I  1.000 I 2.600E  1I     26I*00000000000000000                               I
I  1.100 I 2.000E  1I     20I*0000000000000000                                I
I  1.200 I 1.700E  1I     17I*000000000000000                                 I
I  1.300 I 8.000E  0I      8I*00000000000                                     I
I  1.400 I 5.000E  0I      5I*00000000                                        I
I  1.500 I 5.000E  0I      5I*00000000                                        I
I  1.600 I 2.000E  0I      2I*000                                             I
I  1.700 I 3.000E  0I      3I*00000                                           I
I  1.800 I 4.000E  0I      4I*0000000                                         I
I  1.900 I 4.000E  0I      4I*0000000                                         I
I  2.000 I 4.000E  0I      4I*0000000                                         I
I  2.100 I  .000E  0I      0I                                                 I
I  2.200 I  .000E  0I      0I                                                 I
I  2.300 I 3.000E  0I      3I*00000                                           I
I  2.400 I 2.000E  0I      2I*000                                             I
I  2.500 I 1.000E  0I      1I0                                                I
I  2.600 I 4.000E  0I      4I*0000000                                         I
I  2.700 I 2.000E  0I      2I*000                                             I
I  2.800 I  .000E  0I      0I                                                 I
I  2.900 I 1.000E  0I      1I0                                                I
I  3.000 I  .000E  0I      0I                                                 I
I  3.100 I  .000E  0I      0I                                                 I
I  3.200 I 2.000E  0I      2I*000                                             I
I  3.300 I  .000E  0I      0I                                                 I
I  3.400 I  .000E  0I      0I                                                 I
I  3.500 I  .000E  0I      0I                                                 I
I  3.600 I  .000E  0I      0I                                                 I
I  3.700 I  .000E  0I      0I                                                 I
I  3.800 I  .000E  0I      0I                                                 I
I  3.900 I  .000E  0I      0I                                                 I
I  4.000 I  .000E  0I      0I                                                 I
I  4.100 I  .000E  0I      0I                                                 I
I  4.200 I  .000E  0I      0I                                                 I
I  4.300 I 1.000E  0I      1I0                                                I
I  4.400 I  .000E  0I      0I                                                 I
I  4.500 I  .000E  0I      0I                                                 I
I  4.600 I  .000E  0I      0I                                                 I
I  4.700 I 1.000E  0I      1I0                                                I
I  4.800 I  .000E  0I      0I                                                 I
I  4.900 I  .000E  0I      0I                                                 I
I  5.000 I  .000E  0I      0I                                                 I
I   E  1 I  .000E  0I      0I                                                 I
+--------+----------+-------+------------+-----------+------------+-----------+
    x      Lg(dN/dx)  dN/dx 1.0E+00      1.0E+01     1.0E+02      1.0E+03
                             "0" : No. of Events in Log. scale.
```

Output 3.12 The number of trials distribution

If this distribution has a long tail, this means generation efficiency is low, then the following points should be tested:

(1) The grids determined by `BASES` is not optimized well. If this is the case, try integration again with more sampling points (by setting `NCALL` larger than the current number).

(2) The integrand does not match for the probability distribution in the input file. Check whether the subprograms `USERIN` and `FUNC` are exactly identical to those used in the integration.

(3) The integration could not give a good answer due to unsuitable integration variables for the integrand. In this case, improvement of the kinematics is required.

Chapter 4

How to use GRACE system

The GRACE system was developed on a main frame computer FACOM, but now it is available on UNIX system. The usage of GRACE on FACOM is basically to use JCL in the Batch Job environment. Although an interactive mode is available on an UNIX system, the interpreter of GRACE on UNIX system is still very primitive. It will be improved in near future. We suppose the X-Window system is available on UNIX system, which is used for drawing Feynman graphs on the screen. The UNIX systems, where we have installed and tested GRACE, are SUN SPARC and HP9000/750.

In the first section the usage on UNIX machine is described and the second section is devoted to that on the main frame computer. In the last section how to run on a parallel processor is briefly described.

4.1 Running on UNIX

In this section, we describe how to execute GRACE system on UNIX system. We assume the "GRACEDIR" is a directory where GRACE system is installed.
At first user should add the following statement in the file ".cshrc". [1]

```
setenv GRACEDIR /usr/local/grace
set path=($path $(GRACEDIR)/bin)
```

`$(GRACEDIR)/bin` is a subdirectory where the executable system of GRACE has been installed.
It is recommended to create a new directory for the calculation of one physical process. For the process $e^+e^- \to W^+W^-\gamma$, as an example, we create a directory eewwa and move to the directory as below:

```
grace% mkdir eewwa
grace% cd eewwa
```

Then we start to generate Feynman graphs for the process in the next subsection.

[1] We assume user uses C shell.

4.1.1 Generate Feynman graph

For the graph generation the following two input files are necessary.

1) Definition of physical process

2) Model definition file

Definition of physical process

Specification of defining physical process is described in subsection 3.1.1. When there is the target process in the file `$(GRACEDIR)/data/Index`, the input parameters for the process can be found in a file `dnnnn`, where the number `nnnn` is the process number defined in the file `Index`. For the process $e^+e^- \to W^+W^-\gamma$ we can use the file `d5120`, where the following parameter is saved.

```
* 5120     E+E-  => W+ W- A     TREE
WORDER      3
INITIAL EL  1
INITIAL ELB 1
FINAL   WB  1
FINAL   WBB 1
FINAL   AB  1
END
```

Fig. 4.1 Input file for the process $e^+e^- \to W^+W^-\gamma$

When there is no input-parameter file for the process to be studied, we have to write them by ourselves according to the specification described in subsection 3.1.1.

Model definition file

Two model definition files, "`particle.table`" and "`particle.table0`", are prepared in `GRACE` system as the standard, whose descriptions are given in chapter 6. As the default the file `$(GRACEDIR)/data/particle.table` is used for the model definition file. If user wants to use his own model definition file instead of this default file, then user should specify the input file name (for example "`myparticle.table`") by the environment parameter "`GRACETABLE`" as follows;

```
setenv GRACETABLE /home/graceuser/myparticle.table
```

Execute graph generation

Suppose a file `eewwa.input` is created in the current directory `eewwa` as the input file defining the physical process, then the graph generation procedure starts by typing the command :

```
grace% gengraph eewwa.input
```

Then the graph generator creates a file "`INTBL`" in the current directory, to which the particle table is copied from the model definition file. As an output a file "`OUTDS`" is created in the current directory, where the graph information is written. Also the total number of generated graphs is reported.

4.1.2 Draw Feynman graph

A Feynman graph drawer is initiated by the command:

```
grace% draw
```

Then `Drawer` asks the first and last graph numbers, you want to draw, as follows;

```
GRACE Version 1.0

Feynman Graph Drawer.

* Enter Graph numbers (First, Last) or "/" for all graphs.
```

Knowing that there are 28 graphs in the process $e^+e^- \to W^+W^-\gamma$, by typing

```
1, 28
```

or

```
/
```

then `Drawer` prints the current physical process and asks further

```
*particle.table Electro-Weak and QCD, no Cabbibo Mixing, with Scalar

* 5120    E+ E-  => W+ W- A     TREE

Enter N, M ( N*M graphs in a page )
```

When "`2, 3`" is given as the input then the first six graphs are drawn on the screen by the X-Window. Clicking somewhere on this window, the next six graphs will appear. When all graphs are drawn, `Drawer` will be terminated. In the current version of `Drawer` nothing can be done except for drawing Feynman graphs. It will be improved in future. This procedure uses the model definition file "`INTBL`" and the graph information file "`OUTDS`", which should not be changed.

4.1.3 Generate source code

After the graph generation, FORTRAN source code is generated by typing the command:

```
grace% genfort
```

The procedure `genfort` uses the files `INTBL` and `OUTDS` as input, which should not be changed. The files `xxxxxx.f` and `Makefile` are created as the output in the current directory, where `xxxxxx` corresponds to a name of program components described in section 3.2. For instance, subprogram `SETMAS` appeared in section 3.2 is created in a file `setmas.f`.

Editing FORTRAN source codes

The following program components should be finalized by the user.

1) Initialization routine `USERIN`
   In `USERIN` the following initialization should be done (see subsection 3.5.3).

   – Integration parameters

`NDIM`	The number of dimensions of integral
`NWILD`	The number of wild variables
`XL(`i`)`,`XU(`i`)`	The lower and upper bounds of integration variable `X(`i`)`
`IG(`i`)`	The grid optimization flag for i-th variable
`NCALL`	The number of sampling points per iteration
`ACC1`	The expected accuracy for the grid optimization step
`ITMX1`	The maximum iteration number for the grid optimization step
`ACC2`	The expected accuracy for the integration step
`ITMX2`	The maximum iteration number for the integration step

  - Initialization of histograms and scatter plots
    To let `BASES/SPRING` know the maximum numbers of histograms and scatter plots, `BHINIT` should be called. These numbers are given by a parameter statement in the include file "`inclh.f`".

    The initialization routines `XHINIT` and `DHINIT` should be called for each histogram and scatter plot, respectively. In the generated `USERIN` by `GRACE`, histograms for all integration variables and scatter plots for all combinations of them are required. When one wants to take histograms and scatter plots by his own will, this part should be changed according to the specification in subsection 3.5.5.

  - Initialization of the amplitude calculation
    Since this part is automatically generated by `GRACE`, it needs no change.

  - Initialization of the kinematics by calling `KINIT`
    The subroutine `KINIT` *should be prepared by user* (see section 3.3).

2) Function program of the integrand
   The function program `FUNC` should be made in a complete form.

  - *The kinematics routine* `KINEM` *should be prepared by user*, whose specification is given in subsection 3.3.

  - Filling histograms and scatter plots
    In the generated `FUNC` by `GRACE`, histograms for all integration variables and scatter plots for all combinations of them are to be filled here. When one changed the initialization of histograms and scatter plots, their filling parts should be also changed (see subsection 3.5.5).

3) Include file "`inclh.f`"
   This file is included in the main programs `MAINBS`, `MAINSP` and the subroutine `USERIN`, where the maximum numbers of histograms and scatter plots are set by the parameter statement, as described in sections 3.5.2, 3.5.3 and 3.6.2. *Since these numbers are still open in the generated file* `inclh.f`*, they should be given by user.*

4.1.4 Makefile

The command "`genfort`" also generates the makefile. The libraries `BASES/SPRING`, interface to `CHANEL` and `CHANEL` are stored in the directory `GRACELDIR`. The objects commonly used both in `BASES` and `SPRING` are defined by macro name `OBJS`.

```
SHELL         = /bin/csh
FC            = fort77
#
GRACELDIR     = /usr/local/grace/lib
#
BASESLIB      = bases
CHANELLIB     = chanel
BDUMMLIB      = bdummy
#
OBJS          = userin.o amparm.o \
                func.o   amptbl.o ampsum.o ampord.o \
                usrout.o kinit.o  kinem.o  setmas.o \
                am0001.o am0002.o am0003.o am0004.o \
                am0005.o am0006.o am0007.o am0008.o \
                am0009.o am0010.o am0011.o am0012.o \
                am0013.o am0014.o am0015.o am0016.o \
                am0017.o am0018.o am0019.o am0020.o \
                am0021.o am0022.o am0023.o am0024.o \
                am0025.o am0026.o am0027.o am0028.o
INT           = int
INTOBJ        = mainbs.o
SPRING        = spring
SPOBJS        = mainsp.o spevnt.o spinit.o spterm.o
TEST          = test
TESTOBJ       = test.o
#
all:            $(INT) $(SPRING)
#
$(INT):         $(INTOBJ) $(OBJS) $(GRACELDIR)/lib$(BASESLIB).a \
                $(GRACELDIR)/lib$(CHANELLIB).a
                $(FC) $(INTOBJ) $(OBJS) -o $(INT) -L$(GRACELDIR) \
                -l$(BASESLIB) -l$(CHANELLIB) $(FFLAGS)
$(SPRING):      $(SPOBJS) $(OBJS) $(GRACELDIR)/lib$(BASESLIB).a
                $(FC) $(SPOBJS) $(OBJS) -o $(SPRING) -L$(GRACELDIR) \
                -l$(BASESLIB) -l$(CHANELLIB) $(FFLAGS)
#
test:           $(TEST)
#
$(TEST):        $(OBJS) $(TESTOBJ) $(GRACELDIR)/lib$(BDUMMLIB).a \
                $(GRACELDIR)/lib$(CHANELLIB).a
                $(FC) $(TESTOBJ) $(OBJS) -o $(TEST) -L$(GRACELDIR) \
                -l$(BDUMMLIB) -l$(CHANELLIB) $(FFLAGS)
#
clean:
                 rm -f *.o $(INT) $(SPRING) $(TEST)
```

Source list 4.1 Makefile for HP9000/750

The macro names `INT` and `INTOBJ` define the executable and the object of the main program for the integration, respectively. Similarly the macro names `SPRING`, `SPOBJS`, TEST and TESTOBJ are defined.

4.1.5 Test of the gauge invariance

The main program test.f is used to check the generated amplitudes at a point in the integration volume as described in section 3.4. In the main program subroutines USERIN and FUNC are called, which call the histogram packages. Since the histogram has no meaning in this test, we use dummy library for them stored in the directory GRACELDIR. Thus it is not necessary to comment out the statements in the subprograms FUNC and USERIN for this test, which call relevant histogram routines.

The executable test is created and is executed by the following commands:

```
grace% make test
grace% test
```

An example of output from the test for the process $e^+e^- \to W^+W^-\gamma$ is shown in section 3.4, where the consistency with 14 digits is found between the covariant and unitary gauges. It should be noted that this test does not guarantee a complete gauge invariance even though it could give consistency between the two gauges, since it tests only at a specific point in the phase space. It is recommended to test the gauge invariance at several points in the phase space.

4.1.6 Integration

After preparation of the subprograms USERIN, KINIT, FUNC and KINEM and a successful test of gauge invariance, we can proceed to the numerical integration by BASES. At first we should make the executables for the integration and event generation by typing make.

```
grace% make
```

Then the executables int and spring are created.
For integration the command int is used.

```
grace% int
```

When this is the first time to run the integration, the interpreter asks the job parameters with the following prompt:

```
LOOPF,LOOPL,NPRINT,JFLAG
```

where LOOPF, LOOPL, NPRINT and JFLAG are the first and last loop numbers, print flag and job flag, respectively, described in subsection 3.5.1. These parameters are saved in a file "bases.jobprm" in the current directory. From the next run, the system uses this file instead of asking them. When these parameters are needed to change, the new parameters are to be given in this file.
The output from the integration package BASES is normally printed on a screen. When we want to write the output on a file, bases.output as an example, the redirection of UNIX system can be applied as follows;

```
grace% int > bases.output
```

Before termination of the integration job, BASES writes the probability information on a binary file bases.data, which is used for the event generation.
It is recommended to look at the integration result carefully, especially over the convergency behaviors both for the grid optimization and integration steps. When the accuracy of each iteration fluctuates, iteration by iteration, and, in some case, it jumps up suddenly to a large value compared to the other iterations, the resultant estimate of integral may not be reliable. There are two possible origins of this behavior; one is due to too small sampling points and the other due to an unsuitable choice of the integration variables for the integrand (see subsections 2.7.4 and 3.5.6). An example of output for the process $e^+e^- \to W^+W^-\gamma$ is given in subsection 3.5.6.

4.1.7 Event generation

When the four vectors of generated events are to be written on a file, then this file should be open in MAINSP and the four vectors should be written on the file in SPEVNT. In the case, where some other histogram packages are used for taking histograms of generated events, initialization of these histograms should be done in SPINIT, filling them in SPEVNT and printing them in SPTERM. The specifications of these routines are given in section 3.6.3.

Since the executable spring is created by the make command already, the event generation starts by typing

```
grace% spring
```

Then SPRING reads the probability information from the binary file bases.data and asks the number of events and computing time with the following prompt:

```
Number of events, Computing time ( minutes ) ?
```

The event generation will run until the given number of events are generated or the computing time is exhausted. The reason why we have the computing time limit in the event generation is that the generation loop may have a possibility to get into an infinite loop when some mistakes were made (see subsections 3.5.6 item 8, subsections 3.6.1 and 3.6.2). In order to estimate the computing time for the event generation, it is recommended to use the expected generation time given in the computing time information of `BASES` output.(see section 3.5.6 item 6)
When the kinematics is made of a single-valued function, the subprogram `FUNC` should be identical both in the integration and event generation. But if it is not the case, `FUNC` in the event generation should be modified from that in the integration as described in subsection 3.6.1.
An example of output from `SPRING` is shown in subsection 3.6.4, which consists of the general information, original and additional histograms, scatter plots, and number of trials distribution. From the original histograms we can see how the generated events reproduce those distributions produced by the integration. In the number of trials distribution we can see the generation efficiency. In the example about 80 % of events are generated by the first trial.

4.2 Running on FACOM

`GRACE` system on `FACOM` at KEK is installed in the user-id `MLIB`. As mentioned before the `GRACE` system consists of the graph generation subsystem, source generation subsystem, integration subsystem and event generation subsystem. Under the user `MLIB` there are following four files for user's purpose:

1) `MLIB.GRACE.Vyymmdd.CNTL`
contains several members of `JCL` to generate and to execute the graph generation and source generation subsystems.

2) `MLIB.GRACE.Vyymmdd.DATA`
contains the particle table and example of input data for the graph generation and source generation subsystems.

3) `MLIB.GRACE.Vyymmdd.LOAD`
contains the library load module for the graph generation and source generation subsystems, and for the amplitude calculation.

4) `MLIB.BASES50.LOAD`
contains the library load module for the integration and the event generation subsystems.

`Vyymmdd` in the file names means the version number when they are created.

In the following subsections Job control cards necessary to use `GRACE` system are described.

4.2.1 Graph generation and source code generation

On `UNIX` system, the graph generation, drawing graph and source generation are processed as separate procedures. On `FACOM`, however, they can be processed with a single job by submitting the `JCL` below.

Submission of this `JCL` initiates a catalogued procedure `#GRACE`, which uses the following data sets:

1) `//GENFGR.INTBL DD`
   This defines the model used in the graph generation and source generation steps. When this data set definition is commented out, the default model definition file "`PTCLTBL0`" is used. A beginner is recommended to use this default file without change.

   When user wants to use his own model definition file, its file name should be given in this definition as in the example.

2) `//GENFGR.SYSIN DD`
   This defines the input data for defining physical process described in subsection 3.1.1, which is used in the graph generation.

3) `//GENFGR.OUTDS DD`
   This defines the output file of the graph information, by which Feynman graphs are drawn. This file is also used in the source generation as an input. When this definition is commented out, a work file is allocated for this purpose, which can not use after this job.

4) `//CREATE.NEWDS DD`
   This step creates a new file, "`userid.@.D5120.FORT77`" as an example, in which generated FORTRAN source code is written.

5) `//GENFORT.FT05F001 DD`
   The file name created in the step `//CREATE.NEWDS DD` should be given here again to let the source generator know this name.

By submitting a job by this `JCL`, the structure of the diagrams are generated, Feynman graphs are drawn on the output papers and the program components of Feynman amplitudes are written on an indicated file with several members.

```
//XXXXG JOB CLASS=M,REGION=4096K,MSGLEVEL=1
//JOBPROC DD DSN=MLIB.GRACE.Vyymmdd.CNTL,DISP=SHR
//         EXEC #GRACE
//*-----------------------------------------------------------------
//*        PARTICLE TABLE
//*GENFGR.INTBL DD DSN=MLIB.GRACE.Vyymmdd.DATA(PTCLTBL0),DISP=SHR
//*-----------------------------------------------------------------
//*        INPUT DATA
//GENFGR.SYSIN DD DSN=MLIB.GRACE.Vyymmdd.DATA(D5120),DISP=SHR
//         DD DSN=MLIB.GRACE.Vyymmdd.DATA(DEND),DISP=SHR
//*-----------------------------------------------------------------
//*        OUTPUT DATA OF CREATED GRAPHS
//*GENFGR.OUTDS DD DSN=userid.@.D5120.DATA,
//*         DISP=(NEW,CATLG),UNIT=SYSDA,SPACE=(TRK,(2,2),RLSE),
//*         DCB=(RECFM=FB,LRECL=80,BLKSIZE=3120)
//*-----------------------------------------------------------------
//*        CREATE FILE FOR OUTPUT FORTRAN SOURCE CODE
//CREATE.NEWDS DD DSN=userid.@.D5120.FORT77,
//         DISP=(NEW,CATLG),UNIT=SYSDA,SPACE=(TRK,(4,4,4)),
//         DCB=(RECFM=FB,LRECL=80,BLKSIZE=3120)
//*-----------------------------------------------------------------
//*        CREATE FORTRAN SOURCE CODE
//GENFORT.FT05F001 DD *
userid.@.D5120.FORT77
//
```

Source list 4.2 JCL for the graph generation and source generation

A template of this JCL is stored in the file "MLIB.GRACE.Vyymmdd.CNTL(GENALL)." If graphs are not required to draw, a catalogued procedure #NODRAW is to be used instead of #GRACE.

4.2.2 Generation of library

A member #GENLIB in the output file is automatically generated, where a JCL for generating library for this process is. This library for the amplitude calculation (userid.@.D5120.LOAD for example) will be used for the gauge invariance test, integration and event generation.

4.2.3 Test of the generated source code

A member #TEST, which is a JCL for a testing program, is automatically generated. This JCL is for the execution of main program TEST which is used to check the generated amplitudes at a point in the integration volume. The user is recommended to confirm the gauge invariance and Lorentz frame independence before making the integration by BASES. Before submitting the job, one has to prepare the subroutines KINIT and KINEM and to fix their filenames in the JCL.

The main program TEST calls the subprograms USERIN and FUNC, which all the histogram packages. Since the histograms in this test have no meaning, we use dummy library for them, which is stored in the file MLIB.GRACE.Vyymmdd.PROGS(BDUMMY).

```
//XXXXG JOB CLASS=M
//*     MLIB.GRACE.Vyymmdd
// EXEC FORT7CL,
//      PARM.FORT='OPT(3),GOSTMT,NAME,NONUM,NOSTATIS',
//      PARM.LKED='NOMAP,NOLIST,NCAL,ALIAS'
//FORT.SYSINC DD DSN=userid.@.D5120.FORT77,DISP=SHR
//FORT.SYSIN  DD DSN=userid.@.D5120.FORT77(AMPARM),DISP=SHR
//            DD DSN=userid.@.D5120.FORT77(AMPTBL),DISP=SHR
//            DD DSN=userid.@.D5120.FORT77(AMPSUM),DISP=SHR
//            DD DSN=userid.@.D5120.FORT77(AMPORD),DISP=SHR
//            DD DSN=userid.@.D5120.FORT77(SETMAS),DISP=SHR
//*
//            DD DSN=userid.@.D5120.FORT77(AM0001),DISP=SHR
//            DD DSN=userid.@.D5120.FORT77(AM0002),DISP=SHR
//            DD DSN=userid.@.D5120.FORT77(AM0003),DISP=SHR
//            DD DSN=userid.@.D5120.FORT77(AM0004),DISP=SHR
//            DD DSN=userid.@.D5120.FORT77(AM0005),DISP=SHR
//            DD DSN=userid.@.D5120.FORT77(AM0006),DISP=SHR
//            DD DSN=userid.@.D5120.FORT77(AM0007),DISP=SHR
//            DD DSN=userid.@.D5120.FORT77(AM0008),DISP=SHR
//            DD DSN=userid.@.D5120.FORT77(AM0009),DISP=SHR
//            DD DSN=userid.@.D5120.FORT77(AM0010),DISP=SHR
//            DD DSN=userid.@.D5120.FORT77(AM0011),DISP=SHR
//            DD DSN=userid.@.D5120.FORT77(AM0012),DISP=SHR
//            DD DSN=userid.@.D5120.FORT77(AM0013),DISP=SHR
//            DD DSN=userid.@.D5120.FORT77(AM0014),DISP=SHR
//            DD DSN=userid.@.D5120.FORT77(AM0015),DISP=SHR
//            DD DSN=userid.@.D5120.FORT77(AM0016),DISP=SHR
//            DD DSN=userid.@.D5120.FORT77(AM0017),DISP=SHR
//            DD DSN=userid.@.D5120.FORT77(AM0018),DISP=SHR
//            DD DSN=userid.@.D5120.FORT77(AM0019),DISP=SHR
//            DD DSN=userid.@.D5120.FORT77(AM0020),DISP=SHR
//            DD DSN=userid.@.D5120.FORT77(AM0021),DISP=SHR
//            DD DSN=userid.@.D5120.FORT77(AM0022),DISP=SHR
//            DD DSN=userid.@.D5120.FORT77(AM0023),DISP=SHR
//            DD DSN=userid.@.D5120.FORT77(AM0024),DISP=SHR
//            DD DSN=userid.@.D5120.FORT77(AM0025),DISP=SHR
//            DD DSN=userid.@.D5120.FORT77(AM0026),DISP=SHR
//            DD DSN=userid.@.D5120.FORT77(AM0027),DISP=SHR
//            DD DSN=userid.@.D5120.FORT77(AM0028),DISP=SHR
//*
//LKED.SYSLMOD DD DSN=userid.@.D5120.LOAD,
//            DISP=(NEW,CATLG,DELETE),
//       SPACE=(TRK,(007,2,015),RLSE)
```

Source list 4.3 JCL for making the library for amplitude calculation

```
//XXXXT JOB CLASS=M
//*     MTAK.GRACE.V920605.DATA
// EXEC FORT7CLG,
//      PARM.FORT='OPT(3),GOSTMT,SOURCE,NONUM,NOSTATIS',
//      PARM.LKED='NOMAP,NOLIST'
//FORT.SYSINC DD DSN=userid.@.D5120.FORT77,DISP=SHR
//FORT.SYSIN  DD DSN=userid.@.D5120.FORT77(TEST),DISP=SHR
//            DD DSN=userid.@.D5120.FORT77(FUNC),DISP=SHR
//            DD DSN=userid.@.D5120.FORT77(USERIN),DISP=SHR
//            DD DSN=userid.@.D5120.FORT77(USROUT),DISP=SHR
//*           DD DSN=userid.@.D5120.FORT77(SETMAS),DISP=SHR
//            DD DSN=userid.@.D5120.FORT77(KINIT),DISP=SHR
//            DD DSN=userid.@.D5120.FORT77(KINEM),DISP=SHR
//*
//            DD DSN=MLIB.GRACE.Vyymmdd.PROGS(BDUMMY),DISP=SHR
//LKED.SYSLIB DD
//            DD DSN=userid.@.D5120.LOAD,DISP=SHR
//            DD DSN=MLIB.GRACE.Vyymmdd.LOAD,DISP=SHR
//
```

Source list 4.4 JCL for the gauge invariance test

4.2.4 Numerical integration

JCL for the numerical integration is generated in a member #INT, by which the numerical integration over the phase space cab be performed by BASES. Before integration, the following points are to be checked:

1) Include file INCLH
Set the numbers of histograms and scatter plots in the include file INCLH as described in section 3.5.5.

2) Preparation of the subprograms USERIN and FUNC
Their specifications are given in sections 3.5.3 and 3.5.4.

3) Preparation of the kinematics routines
In the GRACE system, KINIT and KINEM are standard subroutine names for initialization and calculation of the kinematics, respectively. Their specifications are given in section 4.3.

4) Probability information
When event generation is considered, a file for the probability information must be defined in the logical unit number 23. For the first integration, this file ('userid.PROCESS.DATA' as an example) should be created. For the second or later use, this file should be use by allocating it as described in the example of JCL.

If only the result of integration is required, this file can be defined as the dummy.

5) Job parameters
   The job parameters described in section 3.5.1, are given as the data cards.

```
//XXXXI JOB CLASS=V
//*     MLIB.GRACE.Vyymmdd
// EXEC FORT7CLG,
//      PARM.FORT='OPT(3),GOSTMT,SOURCE,NONUM,NOSTATIS',
//      PARM.LKED='NOMAP,NOLIST'
//FORT.SYSINC DD DSN=userid.@.D5120.FORT77,DISP=SHR
//FORT.SYSIN  DD DSN=userid.@.D5120.FORT77(MAINBS),DISP=SHR
//            DD DSN=userid.@.D5120.FORT77(USERIN),DISP=SHR
//            DD DSN=userid.@.D5120.FORT77(FUNC),DISP=SHR
//            DD DSN=userid.@.D5120.FORT77(KINIT),DISP=SHR
//            DD DSN=userid.@.D5120.FORT77(KINEM),DISP=SHR
//            DD DSN=userid.@.D5120.FORT77(USROUT),DISP=SHR
//*
//LKED.SYSLIB DD
//            DD DSN=userid.@.D5120.LOAD,DISP=SHR
//            DD DSN=MLIB.GRACE.Vyymmdd.LOAD,DISP=SHR
//            DD DSN=MLIB.BASES50.LOAD,DISP=SHR
//GO.SYSIN    DD *
   1,1                LOOP MIN AND MAX
  -4                  Print Flag
   0                  Flag
 120.0                CPU Time in Minutes
/*
//GO.FT23F001 DD DUMMY
//*   For the second or later use
//*GO.FT23F001 DD DSN=userid.PROCESS.DATA,DISP=SHR
//*   For the first use
//*GO.FT23F001 DD DSN=userid.PROCESS.DATA,DISP=(NEW,CATLG,DELETE),
//*          SPACE=(TRK,(5,5),RLSE),
//*          DCB=(RECFM=VBS,BLKSIZE=23476)
```

Source list 4.5 JCL for the numerical integration by BASES

4.2.5 Event generation

A JCL for the event generation is also generated by GRACE. Before running the event generation, the following items are considered:

1) Include file INCLH
   The numbers of histograms and scatter plots should be set in the include file INCLH, whose numbers must be greater than or equal to those defined in BASES.

2) Subroutines SPINIT, SPEVNT and SPTERM
   These routines are called by SPRING, whose specifications are given in section 3.6.3.

3) Probability information file
   This file should be prepared by BASES, which is read from the logical unit number 23.

4) Input parameters
   At the beginning of the event generation, the number of events and computing time limit are read from the logical unit number 5.

```
//XXXXS JOB CLASS=V
//*     MLIB.GRACE.Vyymmdd
// EXEC FORT7CLG,
//      PARM.FORT='OPT(3),GOSTMT,SOURCE,NONUM,NOSTATIS',
//      PARM.LKED='NOMAP,NOLIST'
//FORT.SYSINC DD DSN=userid.@.D5120.FORT77,DISP=SHR
//FORT.SYSIN  DD DSN=userid.@.D5120.FORT77(MAINSP),DISP=SHR
//            DD DSN=userid.@.D5120.FORT77(USERIN),DISP=SHR
//            DD DSN=userid.@.D5120.FORT77(FUNC),DISP=SHR
//            DD DSN=userid.@.D5120.FORT77(KINIT),DISP=SHR
//            DD DSN=userid.@.D5120.FORT77(KINEM),DISP=SHR
//            DD DSN=userid.@.D5120.FORT77(SPINIT),DISP=SHR
//            DD DSN=userid.@.D5120.FORT77(SPEVNT),DISP=SHR
//            DD DSN=userid.@.D5120.FORT77(SPTERM),DISP=SHR
//*
//LKED.SYSLIB DD
//            DD DSN=userid.@.D5120.LOAD,DISP=SHR
//            DD DSN=MLIB.GRACE.Vyymmdd.LOAD,DISP=SHR
//            DD DSN=MLIB.BASES50.LOAD,DISP=SHR
//GO.SYSIN    DD *
 10000                 Number of events
   1.0                 CPU Time in Minutes
/*
//GO.FT23F001 DD DSN=userid.PROCESS.DATA,DISP=SHR,LABEL=(,,,IN)
```

Source list 4.6 JCL for the event generation by SPRING

4.3 Running on parallel computers

We have tested the BASES of GRACE system on parallel computers (INTEL iPSC/860, nCUBE 2 and Fujitsu AP1000). In this section, we describe how to execute the numerical calculation. We prepare the BASES library for these parallel computers. In particular we assume we use INTEL iPSC/860, because this machine is most powerful one among these three machines we used. User can use other parallel computers in the similar way.

As a remote host computer for INTEL iPSC/860 the SUN SPARC workstation is used. On this remote host computer, the cross-compiler and various software to use INTEL iPSC/860, are installed. Therefore user only login into this remote machine and can use INTEL iPSC/860.

At the first time user should add the following statement in the .cshrc.

```
setenv GRACEDIR /user/local/grace
set path=($path $(GRACEDIR)/bin)
setenv IPSC_XDEV /usr/ipsc/XDEV/R3.0
set path=($path /usr/ipsc/bin $(IPSC_XDEV)/i860/bin.sun4)
```

$(GRACEDIR)/bin is a subdirectory where the executables for GRACE has been installed. Last 2 lines is required in order to create the proper environment for usage of INTEL iPSC/860 itself.

It is recommended to create a new directory for the calculation of one physical process. For the process $e^+e^- \to W^+W^-\gamma$ a directory eewwa is created and move to the directory:

```
grace% mkdir eewwa
grace% cd eewwa
```

User can perform the following steps using the commands installed on this UNIX machine itself.

- Generation Feynman graph
- Drawing Feynman graph
- Generation source code

See sections 4.1.1, 4.1.2 and 4.1.3.

4.3.1 Command summary for INTEL iPSC/860

In this subsection, we summarize the commands for INTEL iPSC/860.

Command Invocation	Description
`cubeinfo -n`	Displays cube ownership information.
`getcube [-c` *cubename* `] [-t` *cubetype* `]`	Allocate a cube, and make it the currently attached cube.
`load [-c` *cubename* `]` *filename*	Loads a user process into the cube.
`waitcube [-c` *cubename* `]`	Wait for process on node to finish before proceeding.
`killcube [-c` *cubename* `]`	Kill node process.
`relcube [-c` *cubename* `]`	Release a cube.

If user create the executable `int.intel`, user type as follows.

```
grace% getcube -t 16;load int.intel;waitcube;killcube;relcube
```

In this case, user allocates 16 nodes.

4.3.2 Makefile

The command "`genfort`" also generates the makefile. The libraries `BASES/SPRING`, interface to `CHANEL` and `CHANEL` are stored in the directory `GRACELDIR`. The objects commonly used both in `BASES` and `SPRING` are defined by macro name `OBJS`, which is referred in the later description.

```
SHELL         = /bin/csh
INTELHOME     = /usr/ipsc/XDEV/R3.0
FC            = if77
FFLAGS        = -O1 -Knoieee
LD            = ld860
AS            = as860
LIBPATH       = $(INTELHOME)/i860/lib-coff
#
GRACELDIR     = /users/packages/grace/lib/intel
#
BASESLIB      = bases
CHANELLIB     = chanel
BDUMMLIB      = bdummy
#
OBJS          = userin.o amparm.o \
                func.o   amptbl.o ampsum.o ampord.o \
                usrout.o kinit.o  kinem.o  setmas.o \
                am0001.o am0002.o am0003.o am0004.o \
                am0005.o am0006.o am0007.o am0008.o \
                am0009.o am0010.o am0011.o am0012.o \
                am0013.o am0014.o am0015.o am0016.o \
                am0017.o am0018.o am0019.o am0020.o \
                am0021.o am0022.o am0023.o am0024.o \
                am0025.o am0026.o am0027.o am0028.o
```

Source list 4.7 Makefile for INTEL

continue to the next page

```
INT           = int.intel
INTOBJ        = mainbs.o
SPRING        = spring.intel
SPOBJS        = mainsp.o spevnt.o spinit.o spterm.o
TEST          = test.intel
TESTOBJ       = test.o
#
all:            $(INT) $(SPRING)
#
$(INT):         $(INTOBJ) $(OBJS) $(GRACELDIR)/lib$(BASESLIB).a \
                $(GRACELDIR)/lib$(CHANELLIB).a
                $(FC) $(INTOBJ) $(OBJS) -o $(INT) -L$(GRACELDIR) \
                -l$(BASESLIB) -l$(CHANELLIB) $(FFLAGS)
$(SPRING):      $(SPOBJS) $(OBJS) $(GRACELDIR)/lib$(BASESLIB).a
                $(FC) $(SPOBJS) $(OBJS) -o $(SPRING) -L$(GRACELDIR) \
                -l$(BASESLIB) -l$(CHANELLIB) $(FFLAGS)
#
test:           $(TEST)
#
$(TEST):        $(OBJS) $(TESTOBJ) $(GRACELDIR)/lib$(BDUMMLIB).a \
                $(GRACELDIR)/lib$(CHANELLIB).a
                $(FC) $(TESTOBJ) $(OBJS) -o $(TEST) -L$(GRACELDIR) \
                -l$(BDUMMLIB) -l$(CHANELLIB) $(FFLAGS)
#
clean:
                 rm -f *.o $(INT) $(SPRING) $(TEST)

.f.o:
                $(FC) $(FFLAGS) -c $<
```

Source list 4.7 Makefile for INTEL

The macro names `INT` and `INTOBJ` define the executable and the object of the main program for the integration, respectively. Similarly the macro names `SPRING`, `SPOBJS`, `TEST` and `TESTOBJ` are defined.

When the gauge invariance is tested by the generated source, subroutines `USERIN` and `FUNC` are called in the main program, which call the histogram packages. Since the histogram has no meaning in the test, we use dummy library for them, which is stored in the directory `GRACELDIR`. Therefore it is not necessary to change the subprograms `FUNC` and `USERIN` for this test.

4.3.3 Test of the gauge invariance

The main program `test.f` is used to check the generated amplitudes at a point in the integration volume as described in section 3.4. The executable `test` is created and is executed by the following commands:

```
grace% make -f makefile.intel test
grace% getcube -t 1;load intel.test;waitcube;killcube;relcube
```

In this case, user allocates only one node and it is meaningless when use allocate multi nodes because the parallel library is not used for test.

4.3.4 Integration

At first user should make the executables for the integration and event generation by command make.

```
grace% make -f makefile.intel
```

Then the executables int.intel and spring.intel are created.
Before loading process into the node processors, user should prepare one file, that is, "bases.jobprm" described in section 4.1.6. For integration

```
grace% getcube -t 16;load int.intel;waitcube;killcube;relcube
```

It is recommended to open the output file in the main program mainbs.f because terminal I/O takes much time.

```
      if( mynode() .eq. 0 ) then
          open(6,file='bases.output',status='unknown')
      endif
```

In this program, the function "mynode" is a system call of INTEL iPSC/860 itself. Then it returns node ID of calling process (0 to number of $nodes - 1$). In parallel BASES, all output will be generated from only node ID=0.

In this case the output from BASES is written on the file bases.output.

Before termination of the integration job, BASES writes the probability information on the file bases.data, which is used for the event generation.

4.3.5 Event generation

At the first time, user should create the input file where the the number of events and computing time are specified. For instance, we assume this file name is "spring.parm", then user should insert two statements before CALL SPMAIN in the main program mainsp.f.

```
        open(5,file='spring.parm',status='old')
        open(6,file='spring.output',status='unknown')
```

Since the executable `spring.intel` is created by the make command already, the event generation starts by

```
grace% getcube -t 1;load spring.intel;waitcube;killcube;relcube
```

The event generation will run until the given number of events are generated or the computing time is exhausted. Since the spring library is not parallelized, so user should allocate only one node.

Chapter 5

GRACE for a Vector computer

When some calculations are performed repeatedly, a **vector computer** displays its power. Our problem, calculating the numerical value of differential cross section repeatedly very many times to obtain that of cross section, is just the case. It is possible to shorten the execution time of the integration by using the vector computer. In order to make the execution of program fast with the vector computer, a *vectorizable* program has to be written, which is interpreted into *vector* operations by a compiler. When the `REDUCE` system was used for taking trace of γ-matrices, the vectorizable program was used to be made by hand or by using a program package `SPROC` Ref.[9]. However, it is now very easy for us to use the vector computer because `GRACE` system can also produce a vectorizable program.

This chapter is devoted to the `GRACE` system for the vector computer and divided into the following sections:

(1) Generated source code by `GRACE`
`GRACE` system can generate subprograms for `BASES` and `SPRING` in a vectorizable form. They are summarized in section 5.1.

(2) `BASES` on a vector computer
The program structure of vector `BASES` and its usage are described in section 5.2.

(3) `SPRING` on a vector computer
A vector version of `SPRING` is now available. In section 5.3, its algorithm and usage are described.

When the numerical calculation of differential cross section takes very much time, even `SPRING` takes also much time for generating events. In our example of the process $e^+e^- \to W^+W^-\gamma$, consisting of 28 Feynman graphs, generation of 10k events takes a comparable computing time to the integration by `BASES`. We consider a process with more graphs, more computing time is required both for the integration and event generation. This may easily occur in use of `GRACE` system because it generates everything automatically except for the kinematics.

5.1 Generated source code by GRACE

GRACE generates three kinds of source codes for the vector computer. The first is a set of program components for the amplitude calculation, the second for the integration by vector BASES and the third for the event generation by vector SPRING.

Some of these program components are identical to those for the scalar computer, which are indicated by "S" in the following tables, while the program components with "V" are special versions for the vector computer. Since almost all components used in BASES are needed in SPRING, they are appeared in the both items. The program components for the vector BASES and SPRING in the last two items are described later in the relevant sections of this chapter.

The interrelation among these subprograms is shown in figure 5.1, where those subprograms in the **white box** are automatically generated by GRACE, while those in the **shaded box** are already contained in other program packages BASES/SPRING, interface program library to CHANEL, and program package CHANEL.

Fig. 5.1 Relation among the generated subprograms

1) a set of program components for amplitude calculation

`SETMAS`	subroutine	S	Definition of mass and decay width of particle.
`AMPARM`	subroutine	S	Definition of coupling constants and others.
`AMPTBL`	subroutine	V	Call `AMnnnn` to calculate amplitudes.
`AMPSUM`	subroutine	V	Amplitudes are summed after being squared.
`AMnnnn`	subroutine	V	Calculate amplitude of the `nnnn`-th graph, where the number `nnnn` of these routine name is equal to the graph number.
`AMPORD`	subroutine	V	Arranges amplitudes.
`INCL1`	include file	V	Define the common variables for masses, amplitude tables *etc.*
`INCL2`	include file	V	Define the work space for `AMPTBL`.
`INCLVS`	include file	V	Define the vector length.
`TESTV`	main	V	Main program for testing gauge invariance.

Among the above components, only the include file `INCLVS` has to be finalized by the user, while the others can be used as it is.

2) a set of program components for the integration by vector `BASES`

`MAINVB`	main	V	Main program for the integration.
`USERIN`	subroutine	S	Initialization of `BASES` and user's parameters.
`KINIT`	subroutine	S	Initialization of kinematics.
`VBFNCT`	subroutine	V	Calculate the numerical values of differential cross section.
`KINEM`	subroutine	V	To derive particle four momenta from the integration variables.
`USROUT`	subroutine	S	Print the amplitude summary table.
`INCLVS`	include file	V	Define the vector length.
`INCLVB`	include file	V	Define the common for integration variables *etc.*
`INCLH`	include file	S	Define the histogram buffer.

3) a set of program components for the event generation by vector `SPRING`

`MAINVS`	main	V	Main program for the event generation.
`USERIN`	subroutine	S	Initialization of `BASES` and user's parameters.
`KINIT`	subroutine	S	Initialization of kinematics.
`VBFNCT`	subroutine	V	Calculate the numerical values of differential cross section.
`KINEM`	subroutine	V	To derive particle four momenta from the integration variables.
`INCLVS`	include file	V	Define the vector length.
`INCLVB`	include file	V	Define the common for integration variables *etc.*
`INCLH`	include file	S	Define the histogram buffer.
`SPINIT`	subroutine	S	Initialization routine for user's purpose.
`SPEVNT`	subroutine	V	To save four vectors on a file.
`SPTERM`	subroutine	S	Termination routine for user's purpose.

5.1.1 Include file INCL1

A remarkable difference between the vector program and the scalar one is appeared in the include file INCL1, shown in the source list 5.1. All variables relevant to the amplitude calculation are arrays with an additional dimension of the vector length NSIZE in the vector program.
Examples:

Variables for the Scalar		Variables for the Vector	COMMON
AG(0:LAG-1,NGRAPH)	⇒	AG(NSIZE,0:LAG-1,NGRAPH)	/AMGRPH/
PE0001(4)	⇒	PE0001(NSIZE,4)	/AMEXTR/
PPROD(NEXTRN,NEXTRN)	⇒	PPROD(NSIZE,NEXTRN,NEXTRN)	/AMEXTR/

The vector length NSIZE is to be defined in the include file INCLVS, where a statement

```
      PARAMETER ( NSIZE = *** )
```

is. One has to determine NSIZE as described in section 5.2.

```
      PARAMETER (LOUTGO =  2, LINCOM =  1)
      PARAMETER (LANTIP = -1, LPRTCL =  1)
      PARAMETER (LSCALR =  1)
      PARAMETER (LEPEXA =  2, LEPEXW =  3, LEPEXZ =  3, LEPEXG =  2)
      PARAMETER (LEPINA =  4, LEPINW =  4, LEPINZ =  4, LEPING =  3)
      PARAMETER (LEXTRN =  2, LINTRN =  4)
* Table of amplitudes
      PARAMETER (NGRAPH =   28, NEXTRN =   5, LAG =  72)
      PARAMETER (NGRPSQ = NGRAPH*NGRAPH)
      COMMON /AMSLCT/JSELG(NGRAPH), JGRAPH, JHIGGS, JWEAKB
      COMPLEX*16 AG, APROP
      COMMON /AMGRPH/AG(NSIZE,0:LAG-1,NGRAPH),APROP(NSIZE,NGRAPH),
     &               ANCP(NSIZE,NGRAPH),ANSP(0:NGRAPH),
     &               CF(NGRAPH,NGRAPH), IGRAPH(NGRAPH)

* Masses and width of particles
      COMMON /AMMASS/AMWB,AMZB,AMAB,AMXB,AMX3,AMPH,AMLU,AMNE,AMNM,AMNT,
     &               . . . . .
      COMMON /AMGMMA/AGWB,AGZB,AGAB,AGXB,AGX3,AGPH,AGLU,AGNE,AGNM,AGNT,
     &               . . . . .
* Coupling constants
      COMMON /AMCPLC/CZWW       ,CAWW       ,CWWAA      ,CWWZA      ,
     &               . . . . .
```

Source list 5.1 The content of include file INCL1

continue to the next page

```
* Momenta of external particles
      COMMON /AMEXTR/PE0001(NSIZE,4),PE0002(NSIZE,4),
     &               PE0003(NSIZE,4),PE0004(NSIZE,4),
     &               PE0005(NSIZE,4),
     &               PPROD(NSIZE,NEXTRN, NEXTRN)
* Switch of gauge parameters
      COMMON /SMGAUS/IGAU00,IGAUAB,IGAUWB,IGAUZB,IGAUGL
      COMMON /SMGAUG/AGAUGE(0:4)
* Normalization
      COMMON /SMDBGG/FKNORM,FKCALL,NKCALL
* Calculated table of amplitudes
      COMMON /SMATBL/AV, LT, INDEXG
      COMPLEX*16 AV(NSIZE,0:LAG-1)
      INTEGER    LT(0:NEXTRN), INDEXG(NEXTRN)
* For external particles
      COMMON /SMEXTP/
     &    PS0001, EW0001, CE0001,
     &    PS0002, EW0002, CE0002,
     &    EP0003, EW0003,
     &    EP0004, EW0004,
     &    EP0005, EW0005
      REAL*8     PS0001(NSIZE,4,2), EW0001(NSIZE,1)
      COMPLEX*16 CE0001(NSIZE,2,2)
      REAL*8     PS0002(NSIZE,4,2), EW0002(NSIZE,1)
      COMPLEX*16 CE0002(NSIZE,2,2)
      REAL*8     EP0003(NSIZE,4,LEPEXW), EW0003(NSIZE,LEPEXW)
      REAL*8     EP0004(NSIZE,4,LEPEXW), EW0004(NSIZE,LEPEXW)
      REAL*8     EP0005(NSIZE,4,LEPEXA), EW0005(NSIZE,LEPEXA)
```

Source list 5.1 The content of include file INCL1

5.1.2 Subroutine AMPTBL

Another difference between the scalar and the vector versions can be seen, for example, in the subprogram AMPTBL, shown in the source list 5.2.

i) At the first stage of amplitude calculation, the subroutine SMEXTF is called for each external fermion and SMEXTV for each vector particle. They construct tables about the external lines necessary for the succeeding calculation.

   After the first call of SMEXTF, the variable EW0001(1) is set equal to "1", which corresponds to the particle (electron). EW0002(1) is set to "-1" after the second call, which means this particle is the anti-particle (positron).

   Then SMEXTV is called twice with mass of AMWB, which constructs tables of the external $W^\pm$ particles. And finally SMEXTV is called with mass of AMAB for the photon (See section 7.2).

```
      SUBROUTINE AMPTBL(NSAMPL)
** 5120     E+ E-  => W+  W-  A       TREE
      IMPLICIT REAL*8(A-H,O-Z)

      INCLUDE  (INCLVS)
      INCLUDE  (INCL1)
      INCLUDE  (INCL2)
*-----------------------------------------------------------------------
      JGRAPH = 0

*   External lines
      CALL SMEXTF(NSAMPL,LINCOM,AMEL,PE0001,PS0001,CE0001)
      DO 101 J = 1 , NSAMPL
         EW0001(J,1) = LPRTCL
  101 CONTINUE
      CALL SMEXTF(NSAMPL,LOUTGO,AMEL,PE0002,PS0002,CE0002)
      DO 102 J = 1 , NSAMPL
         EW0002(J,1) = LANTIP
  102 CONTINUE
      CALL SMEXTV(NSAMPL,LEPEXW,AMWB,PE0003,EP0003,EW0003,IGAUWB)
      CALL SMEXTV(NSAMPL,LEPEXW,AMWB,PE0004,EP0004,EW0004,IGAUWB)
      CALL SMEXTV(NSAMPL,LEPEXA,AMAB,PE0005,EP0005,EW0005,IGAUAB)

* Graph NO.    1 -   1 (   1)
      IF (JWEAKB.NE.0) THEN
      IF (JSELG(   1).NE.0) THEN

      JGRAPH = JGRAPH + 1
      IGRAPH(JGRAPH) =    1
      CALL AM0001(NSAMPL)
      ENDIF
      ENDIF
  . . . . . . . . . . . .

* Graph NO.   28 -   1 (  28)
      IF (JWEAKB.NE.0) THEN
      IF (JSELG(  28).NE.0) THEN

      JGRAPH = JGRAPH + 1
      IGRAPH(JGRAPH) =   28
      CALL AM0028(NSAMPL)
      ENDIF
      ENDIF
      RETURN
      END
```

Source list 5.2 An example of AMPTBL

ii) The subroutine AMnnnn is called for calculating amplitude of the nnnn-th graph. Since there are 28 Feynman graphs for the process $e^+e^- \to W^+W^-\gamma$, 28 subroutines, AM0001, ..., and AM0028, are called successively.

All subroutines including AMPTBL itself have the variable NSAMPL as the first argument, which is the real vector length determined by BASES/SPRING. When the case of NSIZE $<$ NSAMPL occurs, the program is terminated after printing an error message.

5.2 BASES on a vector computer

Basic idea for vectorizing the integration program BASES is the following:

(A) Initialization

(1) At the beginning of integration the subprogram USERIN is called, where the number of wild variable N_{wild} and that of sampling points per iteration N_{sample}^{given} are given together with other parameters.

(2) From these numbers N_{wild} and N_{sample}^{given} the number of hypercubes N_{cube} is determined by the algorithm described in subsection 2.7. The hypercubes are divided into N_v groups, each of which has $N_{trial} \times N_{cube}/N_v$ $(= N_{sample})$ sampling points per iteration. [1]

One reason for this grouping hypercubes into N_v groups is applicability to a parallel vector computer and another is to keep program size as small as possible on the vector computer.

(B) Calculation of the estimate of integral and its error for one iteration

(1) The random numbers are generated for all sampling points of one group by a vectorized random number generator. As we have N_{sample} sampling points a group and N_{dim} integration variables, $N_{sample} \times N_{dim}$ random numbers are generated.

(2) They are translated into the integration variables, which consist of N_{sample} sets of N_{dim} dimensional variables, where a set corresponds to a sampling point.

(3) The N_{sample} sets of variables are given at the same time to the subprogram VBFNCT, where N_{sample} values of the integrand are calculated by a vectorized program.

(4) Steps (B.1) $\sim$ (B.3) are repeated until the estimates of all groups are finished.

(5) Sums of estimate and variance are taken over all groups and the estimated error is also calculated for the iteration.

(C) The program flows of the grid optimization and integration steps are identical to those in the scalar version.

[1] The maximum number of N_{sample} is taken as that for NSIZE in the include file INCLVS.

Fig. 5.2 Structure of the vector version BASES

Since grouping hypercubes is carried out by the identical algorithm to the parallel computer, the relation between the numbers N_{cube} and N_{node} described in subsection 2.7 is identical to that between the numbers N_{cube} and N_v.

5.2.1 Structure of vector BASES

The program structure of vector version BASES is shown in figure 5.2, where the program components with in the white box generated by GRACE and the others are in the libraries.

From the integration program, the subprograms USERIN, VBFNCT and USROUT are called, which are the interfaces to the user program. The specifications to write these subprograms are described in subsection 5.2.2.

The subroutine VBMAIN controls the global flow of integration, like the initialization stage, the grid optimization step, the integration step and termination stage. This global flow is exactly identical to that for the scalar version described in subsection 3.5.2.

The grid optimization and integration steps are controlled by VBASES, where the distinction between these two steps is done by the job flag as mentioned in subsection 3.5.1.

5.2.2 Subprograms to be prepared

For the integration by BASES the main program MAINVB and the subprograms USERIN, VBFNCT, USROUT should be finalized by the user whose templates are generated by GRACE.

In USERIN, the subroutines SETMAS, AMPARM and KINIT are called, for which the same routines as on the scalar computer can be used. For USROUT the routine used on the scalar computer is also applicable to the vector machine.

The subroutine VBFNCT, however, is not the function subprogram like FUNC on the scalar machine, but must be written by a vectorizable code. Furthermore, it includes the kinematics routine KINEM, which should be vectorized to obtain a high performance of the vector computer.

Main program MAINVB

MAINVB is given as in the form of the source list 5.2.

```
      IMPLICIT REAL*8 (A-H, O-Z)

* Arrays for VBASES
      INCLUDE (INCLVB)
      COMMON /VBCNTO/ KG(NSIZE,MDIM), IA(NSIZE,MDIM)
      COMMOM /VBRAND/ RX(NSIZE), NRN(NSIZE), LAB(NSIZE)
* Arrays for HISTOGRAM and SCAT_PLOT
      INCLUDE (INCLH)
      COMMON /PLOTB/  IBUF( 281*NHIST + 2527*NSCAT + 281 )
      COMMON /VBHIST/ IADX(NSIZE,NHIST+1), IADD(NSIZE,2,NSCAT)
      COMMON /HSTCNT/ IX(NSIZE), IY(NSIZE), IDFX(NSIZE)
* Arrays for CHANEL
      PARAMETER ( NWORD = 167 )
      COMMON /CHWORK/ WORKS(NSIZE,NWORD)
* Control for number of nodes
      COMMON /NINFO/ NODEID, NUMNOD, IPFLAG

      NODEID = 0
      NUMNOD = 16

      CAL VBMAIN( MDIM, NSIZE, X, FX, WGT, IA, KG )

      STOP
      END
```

Source list 5.2 Main program MAINVB

The include file INCLVB is generated automatically by GRACE and its content is as follows:

```
      PARAMETER ( MDIM = 4 )
      INCLUDE (INCLVS)
      COMMON /VBVECT/ WGT(NSIZE), X(NSIZE,MDIM), FX(NSIZE)
```

where the include file INCLVS defines the vector length NSIZE in the following:

```
      PARAMETER ( NSIZE = 302 )
```

The parameter NUMNOD is the number N_v mentioned at the beginning of this section. The reason why this variable name is "number of nodes" is that when we use a parallel vector computer the meaning of this variable becomes really the number of nodes.

The file INCLVS is also created automatically. The random numbers are generated for N_{sample} sampling points by the subprogram VBRNDM, and are transferred into N_{sample} sets of integration variables in routine VBASES. Then they are given to the subprogram VBFNCT, where the values of integrand are calculated at these sets of sampling points.

Subroutine VBFNCT

In the subroutine VBFNCT the values of integrand are calculated for N_{sample} sampling points by a vectorizable code. In the source list 5.3 an example of this routine is given. By comparing with the scalar version in section 3.5.4 we can see the difference between the scalar and vector versions.

```
      SUBROUTINE VBFNCT(NSAMPL)
      IMPLICIT REAL*8(A-H,O-Z)

      PARAMETER ( MXDIM = 50 )
      COMMON / BASE1 / XL(MXDIM),XU(MXDIM),NDIM,NWILD,
     &                 IG(MXDIM),NCALL
      INCLUDE  (INCLVB)
      INCLUDE  (INCL1)
      COMMON /AMREG / MXREG
      COMMON /AMSPIN/ JHS(NEXTRN), JHE(NEXTRN), ASPIN
      REAL*8     ANSO(NSIZE)
* P  : Table of four momenta
* PP : Table of inner products
      REAL*8     P(NSIZE,4,NEXTRN),PP(NSIZE,NEXTRN,NEXTRN)
      REAL*8     YACOB(NSIZE), DFX(NSIZE)
      INTEGER    NREG(NSIZE),JUMP(NSIZE),L(NSIZE)
      COMMON /SP4VEC/ VEC(NSIZE,4,NEXTRN)
      REAL*8 DFT(NSIZE), RN(NSIZE)
*=======================================================================
*                    Initialization
*=======================================================================
      DO 100  I = 1, NSAMPL
        FX(I)   = 0.0D0
        NREG(I) = 1
  100 CONTINUE
*=======================================================================
*                    Kinematics
*=======================================================================
      DO 1000 IREG = 1, MXREG

        CALL KINEM(NSAMPL, NEXTRN, P, PP, YACOB,NREG,IREG,JUMP)

*-----------------------------------------------------------------------
*      Reset the temporal buffer for the region 1
*-----------------------------------------------------------------------
         IF( IREG .EQ. 1 ) THEN
             DO 150 I = 1, NSAMPL
                DFT(I) = 0.D0
  150        CONTINUE
```

Source list 5.3 Subprogram VBFNCT

continue to the next page

```
           DO 180 K = 1, NEXTRN
           DO 180 J = 1, 4
           DO 180 I = 1, NSAMPL
              VEC(I,J,K) = 0.D0
  180      CONTINUE
        ENDIF
*-------------------------------------------------------------------
*      Gather the actual sampling points (MXLOOP) : # of actual S.P.
*-------------------------------------------------------------------
        MXLOOP    = 0
        DO 200 I  = 1, NSAMPL
           DFX(I) = 0.D0
           IF( JUMP(I) .EQ. 0 ) THEN
               MXLOOP = MXLOOP + 1
               L(MXLOOP) = I
           ENDIF
  200   CONTINUE
        IF( MXLOOP .LE. 0 ) GO TO 1000
*-------------------------------------------------------------------
*                 Four momenta of external particles
*-------------------------------------------------------------------
        DO 300 I = 1, 4
        DO 300 J = 1, MXLOOP
*                                     1:  EL-  INITIAL   LPRTCL
           PE0001(J,I)   = P(L(J), I, 1)
*                                     2:  EL+  INITIAL   LANTIP
           PE0002(J,I)   = P(L(J), I, 2)
*                                     3:  WB+  FINAL     LPRTCL
           PE0003(J,I)   = P(L(J), I, 3)
*                                     4:  WB-  FINAL     LANTIP
           PE0004(J,I)   = P(L(J), I, 4)
*                                     5:  AB   FINAL     LPRTCL
           PE0005(J,I)   = P(L(J), I, 5)
  300   CONTINUE
*-------------------------------------------------------------------
*     Inner products of external particles
*-------------------------------------------------------------------

        DO 350 K = 1, NEXTRN
        DO 350 J = 1, NEXTRN
        DO 350 I = 1, MXLOOP
           PPROD(I, J, K) = PP(L(I), J, K)
  350   CONTINUE
```

Source list 5.3 Subprogram VBFNCT

continue to the next page

```
*=====================================================================
*                Amplitude calculation
*=====================================================================

       CALL AMPTBL(MXLOOP)
       CALL AMPSUM(MXLOOP,ANSO)
       DO 400 I = 1, MXLOOP
         DFX(L(I)) = ANSO(I)*YACOB(L(I))*ASPIN
         ANSP(0)   = ANSP(0)  + WGT(L(I))*DFX(L(I))
         FX(L(I))  = FX(L(I)) + DFX(L(I))
  400  CONTINUE

*--- special treatment for SPRING ------------------------------------
*      Save four momenta and probabilities of the region 1
*---------------------------------------------------------------------
        IF( IREG .EQ. 1 ) THEN
            DO 420 K = 1, NEXTRN
            DO 420 J = 1, 4
            DO 420 I = 1, MXLOOP
               VEC(L(I),J,K) = P(L(I),J,K)
  420       CONTINUE
            DO 430 I  = 1, NSAMPL
               DFT(I) = DFX(I)
  430       CONTINUE
        ENDIF

*=====================================================================
*            For histograms and scatter plots
*=====================================================================
       DO  450 I = 1, NDIM
          CALL XVFILL(I, NSAMPL,  X(1,I), DFX )
  450  CONTINUE
       K = 0
       DO  500 I = 1, NDIM - 1
       DO  500 J = I + 1, NDIM
           K = K + 1
           CALL DVFILL(K, NSAMPL, X(1,I), X(1,J), DFX )
  500  CONTINUE
*=====================================================================
*            For summary tables
*=====================================================================
       DO 600  IGR = 1, JGRAPH
       DO 600  I   = 1, MXLOOP
         ANSP(IGR) = ANSP(IGR)
     .               + WGT(L(I))*YACOB(L(I))*ASPIN*ANCP(L(I),IGR)
  600  CONTINUE
 1000 CONTINUE
```

Source list 5.3 Subprogram VBFNCT

continue to the next page

```
      DO 700  I = 1, NSAMPL
         NKCALL = NKCALL + 1
         IF( NKCALL .GT. 10000 ) THEN
             NKCALL = NKCALL - 10000
             FKCALL = FKCALL + 10000
         ENDIF
  700 CONTINUE
*=== Special treatment for SPRING ( in two valued function case ) ======
*        Put the final 4 vectors into the arrays VEC()
*=======================================================================
      CALL VBRNDM( 1, NSIZE, NSAMPL, RN )

      DO 800 I = 1, NSAMPL
         L(I)  = 0
  800 CONTINUE

      MXLOOP   = 0
      DO 820 I = 1, NSAMPL
         IF( FX(I) .GT. 0.D0 ) THEN
             IF( DFT(I)/FX(I) .LT. RN(I)) THEN
                 MXLOOP = MXLOOP + 1
                 L(MXLOOP) = I
             ENDIF
         ENDIF
  820 CONTINUE

      DO 850 K = 1, NEXTRN
      DO 850 J = 1, 4
      DO 850 I = 1, MXLOOP
         VEC(L(I),J,K) = P(L(I),J,K)
  850 CONTINUE

      RETURN
      END
```

Source list 5.3 Subroutine VBFNCT

The structure of the subprogram VBFNCT is as follows:

(1) The number N_{sample} is given by the argument of the subprogram and the numerical values of the integration variables are given by X(NSIZE,MDIM) in common /VBVECT/. The numerical values of integrand are to be stored in FX(NSIZE). The other variables in the common /VBVECT/ are used in BASES and should not be alter anywhere. The content of the include file INCLVB is as follows:

```
      PARAMETER ( MDIM  = 4 )
      PARAMETER ( NSIZE = 302 )
      COMMON /VBVECT/ WGT(NSIZE), X(NSIZE,MDIM), FX(NSIZE)
```

(2) The variables or arrays of `P`, `PP`, `YACOB`, `NREG` and `JUMP` have an additional dimension of the vector length `NSIZE`.

Examples:

Variables for the scalar		Variables for the vector
`P(4,NEXTRN)`	$\Rightarrow$	`P(NSIZE,4,NEXTRN)`
`YACOB`	$\Rightarrow$	`YACOB(NSIZE)`
`NREG`	$\Rightarrow$	`NREG(NSIZE)`

(3) In the subprogram `KINIT`, the multiplicity of kinematics function `MXREG` should be set (See section 3.3.1). For our example of the process $e^+e^- \to W^+W^-\gamma$, as we prepare a single valued function as the kinematics, then we set `MXREG` equal to "1". Furthermore, the values of integrand should be cleared and the variable `NREG` should be initialized to be "1".

```
      DO 100 I   = 1, NSAMPL
         FX(I)   = 0.0D0
         NREG(I) = 1
  100 CONTINUE
```

(4) **Kinematics and the gathering the points.**
In the calculation of kinematics, there may be those sampling points which are outside the kinematical boundary or some detector acceptance. The integrand should not be calculated for these points. In such a case, the "*gathering and scattering*" method is recommended to use. To gather those sampling points, which are in the boundary, the "*list vector*" is used. In the "*list vector*" `L`(i), where the pointers only for those sampling points, which are in the boundary or the flag `JUMP`(i) $= 0$, are stored and the kinematical variables, like four momenta, are gathered into the first `MXLOOP` of an array `PEnnnn(I,J)`, as shown below.

```
* Calculation of kinematics for NSAMPL points at the same time

      CALL KINEM( NSAMPL, NEXTERN, P, PP, YACOB, NREG, IREG, JUMP)

      MXLOOP = 0
      DO 200 I = 1, NSAMPL
         DFX(I) = 0.0D0
         IF( JUMP(I) .EQ. 0 ) THEN
             MXLOOP    = MXLOOP + 1
             L(MXLOOP) = I
         ENDIF
  200 CONTINUE

      DO 300 I = 1, 4
      DO 300 J = 1, MXLOOP
         PE0001(J,I) = P(L(J),I,1)
             . . . . . . . .

         PE0005(J,I) = P(L(J),I,5)
      300 CONTINUE
```

It is noted that the number of arguments of subroutine KINEM is different from that for the scalar version.

(5) **Calculation of amplitudes**

By gathering the sampling points in the arrays PEnnnn() and PPROD(), an efficient amplitude calculation is realized.

(6) **Scatter the gathered points**

To return the values of integrand to BASES, the gathered points should be scattered in the order of the given sampling points. For this purpose we use the list vector L(i), prepared in step (4).

```
          DO 400 I = 1, MXLOOP
             DFX(L(I))= ANSO(I)*YACOB(L(I))*ASPIN
             ANSP(0)  = ANSP(0) + WGT(L(I))*DFX(L(I))
             FX(L(I)) = FX(L(I)) + DFX(L(I))
      400 CONTINUE
```

(7) **Histogramming**

To fill the histogram (scatter plot), the subroutine XVFILL (DVFILL) is called as in the following way. The filling routines on the vector computer are different from those on the scalar computer. The variables X(1,I) and X(1,J) are arrays, which should be defined somewhere in VBFNCT and should not be the gathered variables. An array DFX is assumed to be the storage of the values of integrand for the scattered points as defined in the step (6).

```
          CALL XVFILL( ID#, NSAMPL, X(1,I), DFX)
          CALL DVFILL( ID#, NSAMPL, X(1,I), X(1,J), DFX)
```

It is noted that the array DFX is cleared at the step (3). Then those sampling points which are out of kinematical boundary contribute to the histogram with weight zero.

Subroutines for Kinematics

GRACE generates the program source code for the amplitude squared but does not for the kinematics part. Therefore the kinematics routine should be vectorized by oneself. Vectorization of a program is quite simple. The statement

```
      A = B + C
```

is vectorized by rewriting as

```
      DO 100 I = 1, L
         A(I)  = B(I) + C(I)
  100 CONTINUE
```

where L is the vector length. However, one should note the following points. Consider an addition operator A_ν, which operates on two n-dimensional vectors x, y and returns an n-dimensional vector z whose ith element has the value of the sum of the ith elements of x and y:

$$z = A_\nu(x, y), \quad \text{where} \quad z_i = x_i + y_i, \quad \text{for } i = 1, ..., n.$$

The parallel operation for each element does not always give the correct result even though the code is vectorized. In the calculation of $x_i = a + x_{i-2}$, for example, x_3 should be determined before $x_5 = a + x_3$ is calculated. Thus there should be no mixing between input and output data in a vectorized program.

In the source list 5.4, an example of KINEM for the process $e^+e^- \to W^+W^-\gamma$ is shown, which is a vectorized code of that for the scalar machine given in section 3.3.

```
      SUBROUTINE KINEM(NSAMPL, NEXTRN, PE, PP, YACOB, NREG, IREG, JUMP)
      IMPLICIT REAL* 8(A-H,O-Z)
      INCLUDE (INCLVB)
      INTEGER NEXTRN
      REAL*8  PE(NSIZE,4,NEXTRN), PP(NSIZE,NEXTRN,NEXTRN)
      REAL*8  YACOB(NSIZE)
      INTEGER NREG(NSIZE), IREG
      INTEGER JUMP(NSIZE)
      COMMON /AMCNST/ PI, PI2, RAD, GEVPB, ALPHA
* Masses and width of particles
      COMMON /AMMASS/AMWB,AMZB,AMAB,AMXB,AMX3,AMPH,AMLU,AMNE,AMNM,AMNT,
     &              AMLD,AMEL,AMMU,AMTA,AMQU,AMUQ,AMCQ,AMTQ,AMQD,AMDQ,
     &              AMSQ,AMBQ,AMCP,AMCM,AMCZ,AMCA,AMGL
      COMMON /AMGMMA/AGWB,AGZB,AGAB,AGXB,AGX3,AGPH,AGLU,AGNE,AGNM,AGNT,
     &              AGLD,AGEL,AGMU,AGTA,AGQU,AGUQ,AGCQ,AGTQ,AGQD,AGDQ,
     &              AGSQ,AGBQ,AGCP,AGCM,AGCZ,AGCA,AGGL
      COMMON / ENRGY / S,W,E,P,P1P2,FACT
      COMMON / TRNSF / YACO,EPSP,AP,XLOG
      COMMON / KCUTS / RMN,RMX,ETH
      COMMON / ACUTS / DELCUT,DLTCSG,DLTCSO,CSMX,CSMN
      COMMON / MASS1 / EM,WM
      COMMON / MASS2 / EM2,WM2
*------------------------------------------------------- kinem-2
*                   Kinematics for the process
*              e-(P1) + e+(P2) ----> W-(Q1) + W+(Q2) + gamma(R)
*
*  (1)  Frame of reference :
*         (a)   Photon is along the z-axis.
*         (b)   Initial  e+  is in the x-z plane.
```

Source list 5.4 An example of a vectorized code of KINEM

continue to the next page

```
*  (2)  Definition of variables :
*         (a)   Polar angle of  e+  is  CSG.
*         (b)   Photon energy is  R.
*         (c)   Polar angle of  W+  is  CSO  and
*               azimuthal angle is  PHI.
*         (d)   Energies of  W-  and  W+  are  Q10 and Q20.
*         (e)   Angle between  e+  and  W+  is  CSTH.
*  (3)  Variable sequence :       R  --->  CSG  --->  Q20  --->  PHI
*-------------------------------------------------------------------
      REAL*8 R(NSIZE),   DR(NSIZE),  CSG(NSIZE),  SNG(NSIZE), D1(NSIZE)
      REAL*8 D2(NSIZE),  ER(NSIZE),  Q20MX(NSIZE),Q20MN(NSIZE)
      REAL*8 DQ20(NSIZE),Q20(NSIZE), Q10(NSIZE),  Q2(NSIZE),  Q1(NSIZE)
      REAL*8 CSO(NSIZE), SNO(NSIZE), CSPHI(NSIZE),SNPHI(NSIZE)
      REAL*8 CSTH(NSIZE),D3(NSIZE),  D4(NSIZE),   P1Q2(NSIZE)
      REAL*8 P2Q2(NSIZE),Q1Q2(NSIZE),P1Q1(NSIZE), P2Q1(NSIZE)
*--------------------------- Entry point -------------------------------
      NCNT    = 0
      DO 10 I = 1, NSAMPL
         IF( IREG .LE. NREG(I)) THEN
             JUMP(I) = 0
         ELSE
             NCNT = NCNT + 1
             JUMP(I) = 1
         ENDIF
   10 CONTINUE
      IF( NCNT .GE. NSAMPL ) RETURN
*==================================================================
*-------------------------------------------------------------  R
      DO 100 I = 1, NSAMPL
         IF( JUMP(I) .EQ. 0 ) THEN
             RR    =   RMX/RMN
             R(I)  = RMN*RR**X(I,1)
             DR(I) = LOG( RR )/R(I)
         ENDIF
  100 CONTINUE
C-------------------------------------------------------------   CSG
      DO 150 I = 1, NSAMPL
         IF( JUMP(I) .EQ. 0 ) THEN
             ZZZ    = EXP( 2.D0*XLOG*( X(I,2) - 0.5D0 ) )
             CSG(I) = ( ZZZ - 1.D0 )/( ZZZ + 1.D0 )*AP
             SNG(I) = SQRT(( 1.D0 - CSG(I) )*( 1.D0 + CSG(I) ))
             D2K    = AP*( 2.D0/( 1.D0 + ZZZ ) )
             D1K    = ZZZ*D2K
             D1(I)  = R(I)*P*D1K
             D2(I)  = R(I)*P*D2K
         ENDIF
  150 CONTINUE
```

Source list 5.4 An example of a vectorized code of KINEM

continue to the next page

```
C---------------------------------------------------------- Q10, Q20
      DO 200 I = 1, NSAMPL
         IF( JUMP(I) .EQ. 0 ) THEN
             CSOMX    = 1.D0 - DLTCSO
             RCSO2    = ( R(I)*CSOMX )**2
             WR       = W - R(I)
             ER(I)    = E - R(I)
             U        = WR**2 - RCSO2
             V        = W*WR*ER(I)
             D        = RCSO2*( ( W*ER(I) )**2 - WM2*U )
             SQD      = SQRT( D )
             Q20MX(I) = ( V + SQD )/U
             Q20MN(I) = ( V - SQD )/U
         ENDIF
  200 CONTINUE
      DO 250 I = 1, NSAMPL
         IF( JUMP(I) .EQ. 0 ) THEN
             A  =   E - Q20MX(I)
             B  =   E - Q20MN(I)
             CA  =   Q20MX(I) - ER(I)
             CB  =   Q20MN(I) - ER(I)
             RX  =   B*CA/( CB*A )
             DQ20(I) = LOG( RX )/( S*R(I) )
             ZZZ  =   A/CA*RX**X(I,3)
             XXX  =   R(I)*ZZZ/( 1.D0 + ZZZ )
             Q20(I) = E - XXX
             Q2(I) =  SQRT(( Q20(I) - WM )*( Q20(I) + WM ))
             Q10(I) = W - Q20(I) - R(I)
             Q1(I) =  SQRT(( Q10(I) - WM )*( Q10(I) + WM ))
         ENDIF
  250 CONTINUE
      DO 300 I = 1, NSAMPL
         IF( JUMP(I) .EQ. 0 ) THEN
             IF( Q20(I) .LT. ETH  .OR.  Q10(I) .LT. ETH  ) THEN
                 JUMP(I) = 1
             ENDIF
         ENDIF
  300 CONTINUE
C-------------------------------------------------------------   CSO
      DPHI  =   2.D0*PI
      DO 350 I = 1, NSAMPL
         IF( JUMP(I) .EQ. 0 ) THEN
             CSO(I) = ( W*( E - R(I) - Q20(I) ) + R(I)*Q20(I) )
     .                 /( Q2(I)*R(I) )
             SNO(I) = SQRT(( 1.D0 - CSO(I) )*( 1.D0 + CSO(I) ))
C-------------------------------------------------------------   PHI
             CSPHI(I) = COS( DPHI*X(I,4) )
             SNPHI(I) = SIN( DPHI*X(I,4) )
         ENDIF
  350 CONTINUE
```

Source list 5.4 An example of a vectorized code of KINEM

continue to the next page

```
C--------------------------------------------------------------   CSTH
      DO 400 I = 1, NSAMPL
         IF( JUMP(I) .EQ. 0 ) THEN
             CSTH(I) = CSO(I)*CSG(I) + SNO(I)*SNG(I)*CSPHI(I)
             IF(  CSTH(I) .GT. CSMX .OR.  CSTH(I) .LT. CSMN ) THEN
                  JUMP(I) = 1
             ENDIF
         ENDIF
  400 CONTINUE
C--------------------------------------------------------------   CS01
      DO 450 I = 1, NSAMPL
         IF( JUMP(I) .EQ. 0 ) THEN
             CS01  =   - ( R(I) + Q2(I)*CSO(I) )/Q1(I)
             IF( CS01  .GT.  1.D0 - DLTCSO  ) THEN
                 JUMP(I) = 1
             ENDIF
         ENDIF
  450 CONTINUE
C-----------------------------------------------------------------------
      DO 500 I = 1, NSAMPL
         IF( JUMP(I) .EQ. 0 ) THEN
             CSQ  =   - ( R(I)*CSG(I) + Q2(I)*CSTH(I) )/Q1(I)
             IF( CSQ  .GT.  CSMX  .OR.  CSQ  .LT.  CSMN  ) THEN
                 JUMP(I) = 1
             ENDIF
         ENDIF
  500 CONTINUE
      DO 550 I = 1, NSAMPL
         IF( JUMP(I) .EQ. 0 ) THEN
             COSDEL = ( Q20(I)*Q10(I) - W*( Q20(I)+Q10(I) )+ E*W + WM2 )
     &                /( Q2(I)*Q1(I) )
             IF(  COSDEL.  GT.  DELCUT  ) THEN
                 JUMP(I) = 1
             ENDIF
         ENDIF
  550 CONTINUE

C--------------------------------------------------------  invariants
      DO 600 I = 1, NSAMPL
         IF( JUMP(I) .EQ. 0 ) THEN
           EPSQ  =   WM2/( Q20(I) + Q2(I) )
             D3(I) = R(I)*E/ER(I)*( EPSQ + Q2(I)*( 1.D0 + CSO(I) ) )
             D4(I) = R(I)*( EPSQ + Q2(I)*( 1.D0 - CSO(I) ) )
           P1Q2(I) = E*Q20(I) + P*Q2(I)*CSTH(I)
           P2Q2(I) = E*Q20(I) - P*Q2(I)*CSTH(I)
           Q1Q2(I) = W*( E - R(I) ) - WM2
           P1Q1(I) = EM2 + P1P2 - D1(I) - P1Q2(I)
           P2Q1(I) = EM2 + P1P2 - D2(I) - P2Q2(I)
         ENDIF
  600 CONTINUE
```

Source list 5.4 An example of a vectorized code of KINEM

continue to the next page

```
*=====================================================================
*     Table of four momenta.

* PE(I, J) : I = 1 -> X, 2 -> Y, ... 4 -> energy, of J-th particle.
      DO 650 I = 1, NSAMPL
         IF( JUMP(I) .EQ. 0 ) THEN
*                              2:        EL+  INITIAL   LANTIP
             PE(I,1,2) = P*SNG(I)
             PE(I,2,2) = 0.0D0
             PE(I,3,2) = P*CSG(I)
             PE(I,4,2) = E
*                              1:        EL-  INITIAL   LPRTCL
             PE(I,1,1) = -PE(I,1,2)
             PE(I,2,1) = -PE(I,2,2)
             PE(I,3,1) = -PE(I,3,2)
             PE(I,4,1) = PE(I,4,2)
         ENDIF
  650 CONTINUE
      DO 700 I = 1, NSAMPL
         IF( JUMP(I) .EQ. 0 ) THEN
*                              3:        WB+  FINAL     LPRTCL
             PE(I,1,3) = Q2(I)*SNO(I)*CSPHI(I)
             PE(I,2,3) = Q2(I)*SNO(I)*SNPHI(I)
             PE(I,3,3) = Q2(I)*CSO(I)
             PE(I,4,3) = Q20(I)
*                              5:        AB   FINAL     LPRTCL
             PE(I,1,5) = 0.0D0
             PE(I,2,5) = 0.0D0
             PE(I,3,5) = R(I)
             PE(I,4,5) = R(I)
         ENDIF
  700 CONTINUE
      DO 750 I = 1, NSAMPL
         IF( JUMP(I) .EQ. 0 ) THEN
*                             4:        WB-  FINAL     LANTIP
             PE(I,1,4) = PE(I,1,1)+PE(I,1,2)-PE(I,1,5)-PE(I,1,3)
             PE(I,2,4) = PE(I,2,1)+PE(I,2,2)-PE(I,2,5)-PE(I,2,3)
             PE(I,3,4) = PE(I,3,1)+PE(I,3,2)-PE(I,3,5)-PE(I,3,3)
             PE(I,4,4) = Q10(I)
         ENDIF
  750 CONTINUE
```

Source list 5.4 An example of a vectorized code of KINEM

continue to the next page

```
C---------------------------------------------------------  invariants
*     PP(I,J) = inner product between PE(*,I) and PE(*,J)

      DO 800 I = 1, NSAMPL
         IF( JUMP(I) .EQ. 0 ) THEN
             PP(I,1,1) = EM2
             PP(I,1,2) = P1P2
             PP(I,1,3) = P1Q2(I)
             PP(I,1,4) = P1Q1(I)
             PP(I,1,5) = D1(I)

             PP(I,2,1) = P1P2
             PP(I,2,2) = EM2
             PP(I,2,3) = P2Q2(I)
             PP(I,2,4) = P2Q1(I)
             PP(I,2,5) = D2(I)
         ENDIF
  800 CONTINUE
      DO 850 I = 1, NSAMPL
         IF( JUMP(I) .EQ. 0 ) THEN
             PP(I,3,1) = P1Q2(I)
             PP(I,3,2) = P2Q2(I)
             PP(I,3,3) = WM2
             PP(I,3,4) = Q1Q2(I)
             PP(I,3,5) = D4(I)

             PP(I,4,1) = P1Q1(I)
             PP(I,4,2) = P2Q1(I)
             PP(I,4,3) = Q1Q2(I)
             PP(I,4,4) = WM2
             PP(I,4,5) = D3(I)

             PP(I,5,1) = D1(I)
             PP(I,5,2) = D2(I)
             PP(I,5,3) = D4(I)
             PP(I,5,4) = D3(I)
             PP(I,5,5) = 0.0D0
         ENDIF
  850 CONTINUE
C--------------------------------------------------------  Jacobian
      DO 900 I = 1, NSAMPL
         IF( JUMP(I) .EQ. 0 ) THEN
             YACOB(I)  =  FACT*DR(I)*(YACO*D1(I)*D2(I))
     .                    *(DQ20(I)*D3(I)*D4(I))*DPHI/2.D0
         ENDIF
  900 CONTINUE

      RETURN
      END
```

Source list 5.4 An example of a vectorized code of KINEM

The structure of this example is as follows:

(1) The number of sampling point NSAMPL is given in the arguments of subroutine KINEM. The arguments NREG and JUMP are now arrays.

(2) The variables, used several times in this program, like R, DR *etc.*, are now arrays to keep their values.

(3) At the beginning, the array JUMP(i) is initialized to be "0", when the condition IREG $\leq$ NREG(i) is satisfied. Otherwise, JUMP(i) is set to "1".

(4) The kinematics is calculated only for those sampling points, each of which has JUMP(i) $= 0$.

(5) When the i-th sampling point is out of the kinematical boundary, then the value of JUMP(i) is set equal to "1".

```
          DO 300 I = 1, NSAMPL
             IF( JUMP(I) .EQ. 0 ) THEN
                IF( Q20(I) .LT. ETH .OR. Q10(I) .LT. ETH ) THEN
                   JUMP(I) = 1
                ENDIF
             ENDIF
      300 CONTINUE
```

(6) At last, the four momenta of external particles PEnnnn( ) and their inner products PP( ) are calculated only for the JUMP(i) $= 0$ sampling points.

5.3 Event generation

For a long time authors have never tried to develop a vector version of SPRING due to such a prejudice that the vector computer is not suitable for the Monte Carlo event generation. One reason, why we dare to make it, is to shorten the execution time of the event generation. Another is the following. When we generate four vectors of very complicated process, at first we integrate the differential cross section by BASES on the vector computer, at second copy the probability information from the vector computer to the scalar computer, and then generate four vector by SPRING on the scalar computer. This procedure is not only complicated, but we also have to have two kinds of subprograms FUNC and VBFNCT. This situation makes easily a problem of inconsistency between FUNC and VBFNCT. When a vector version of SPRING is available, we are free from such problems.

5.3.1 Event generation algorithm on a vector computer

In figure 5.3 the algorithm of event generation on a vector computer is shown, where the NSIZE objects of array IC(i) , i = 1, NSIZE, are considered, each of which corresponds

to a candidate of generating event. Here, the number NSIZE is identical to that in the include file INCLVS and is the vector length or the length of the inner most do-loop. Suppose we have integrated a distribution function by BASES to generate events and have written the probability information on a output file.
Then,

(1) Sample NSIZE hypercubes according to the probability information and store the hypercube numbers in the array IC.

(2) Sample NSIZE sampling points, where each point belongs to each sampled hypercube.

(3) Calculate the numerical values of the distribution function at these sampling points.

(4) Generate NSIZE uniform random numbers, store them in a array RX( ), test the values of distribution function by them, and set those hypercube numbers equal to zero whose sampling points are accepted by the test.

    The test condition is :

```
          DO 100 I = 1, NSIZE
             IF( FX(I)/FMX(IC(I)) .LT. RX(I) ) THEN
                IC(I) = 0
             ENDIF
      100 CONTINUE
```

    where FX(*i*) and FMX(IC(*i*)) are the numerical value of the distribution function for the hypercube of *i*-th object and the maximum value for the object, respectively.

(5) Write four vectors of those events on a output file, whose hypercube numbers were set equal to zero in the step (4).

(6) Only the accepted sampling points are filled into the histograms in the subprogram VSHUPD by using the flag array IC(*i*).

(7) Sample hypercubes only for those objects whose hypercube numbers were set to zero in the step (4) and store the new hypercube numbers into them.

(8) Go to (2) unless the number of generated events exceeds the given number.

In this algorithm almost all calculation can be fully vectorized.

Fig. 5.3 Algorithm of the event generation on a vector computer

Fig. 5.4 Program structure of the vector `SPRING50`

The program structure of the vector version `SPRING` is shown in figure 5.4, where the subprograms in the `white box` are generated by `GRACE`.

In the vector version the subprogram `VSMAIN` controls program flow as follows:

(A) Initialization

The subprograms `BSREAD, USERIN, SPINIT` and `SHRSET` are called, whose specifications are identical to those for the scalar version.

(B) Event generation loop
The event generation loop is controlled by the routine VSPRNG, whose algorithm is described above. The four vector of generated events are to be calculated in SPEVNT only for those points, and to be written on a file, whose flag IC is zero (*i.e.* only the accepted points are written). In VSHUPD, only the accepted points are automatically filled into the histograms by using the flag array IC(i).

(C) Termination of process by printing the general information and histograms *etc.*

5.3.2 Subroutine to be prepared

The specifications of the subprograms USERIN and VBFNCT are identical to those for the vector BASES. The subprograms SPINIT and SPTERM are also identical to the scalar version. The difference appears only in the subprogram SPEVNT. In the scalar version SPEVNT has no argument, but the vector version of SPEVNT has the array as an argument.

An example is shown in the source list 5.5, where the polar angle distribution of the photon in the process $e^+e^- \to W^+W^-\gamma$ is filling into an additional histogram. Its function is exactly identical to that of the example in subsection 3.6.3. The argument is the flag array IC and its size NSAMPL. We can see that only the i-th point of array VEC(i,*,*) is filled into the histogram, whose flag IC(i) is equal to zero.

The output from the vector SPRING is exactly identical to that from the scalar version.

```
      SUBROUTINE SPEVNT( NSAMPL, IC )
      IMPLICIT REAL*8 ( A-H, O-Z)
      INTEGER IC(NSAMPL)
*
      PARAMETER (NGRAPH =   28, NEXTRN =   5, LAG =  72)
      INCLUDE (INCLVS)
      COMMON /SP4VEC/ VEC(NSIZE,4,NEXTRN)
      COMMON / AMCNST / PI, PI2, RAD, GEVPB, ALPHA
      DIMENSION PP(NSIZE),TH(NSIZE)

      DO 100 I = 1, NSAMPL
         IF( IC(I) .EQ. 0 ) THEN
             PP(I) = SQRT((VEC(I,1,1)**2+VEC(I,2,1)**2+VEC(I,3,1)**2)
     .                   *(VEC(I,1,5)**2+VEC(I,2,5)**2+VEC(I,3,5)**2))
         ENDIF
  100 CONTINUE
```

Source list 5.5 An example of SPEVNT for the vector version

continue to the next page

```
      DO 200 I = 1, NSAMPL
         IF( IC(I) .EQ. 0 ) THEN
             CS    =( VEC(I,1,1)*VEC(I,1,5)+VEC(I,2,1)*VEC(I,2,5)
     .                 +VEC(I,3,1)*VEC(I,3,5) )/PP(I)
             TH(I) = ACOS(CS)*180.D0/PI
         ENDIF
  200 CONTINUE

      CALL XVFILL( 5, NSAMPL, TH, PP )

      RETURN
      END
```

Source list 5.5 An example of SPEVNT for the vector version

```
//XXXXG JOB CLASS=M,REGION=4096K,MSGLEVEL=1
//JOBPROC DD DSN=Mxxx.GRACE.Vyymmdd.PROGS,DISP=SHR
//         EXEC #GRACEV
//*-----------------------------------------------------------------
//*        PARTICLE TABLE
//*GENFGR.INTBL DD DSN=Mxxx.GRACE.Vyymmdd.DATA(PTCLTBL0),DISP=SHR
//*-----------------------------------------------------------------
//*        INPUT DATA
//GENFGR.SYSIN DD DSN=Mxxx.GRACE.Vyymmdd.DATA(D5120),DISP=SHR
//         DD DSN=Mxxx.GRACE.Vyymmdd.DATA(DEND),DISP=SHR
//*-----------------------------------------------------------------
//*        OUTPUT DATA OF CREATED GRAPHS
//*GENFGR.OUTDS DD DSN=userid.@.D5120.DATA,
//*         DISP=(NEW,CATLG),UNIT=SYSDA,SPACE=(TRK,(2,2),RLSE),
//*         DCB=(RECFM=FB,LRECL=80,BLKSIZE=3120)
//*-----------------------------------------------------------------
//*        CREATE FILE FOR OUTPUT FORTRAN SOURCE CODE
//CREATE.NEWDS DD DSN=userid.@.D5120.FORT77,
//         DISP=(NEW,CATLG),UNIT=SYSDA,SPACE=(TRK,(4,4,4)),
//         DCB=(RECFM=FB,LRECL=80,BLKSIZE=3120)
//*-----------------------------------------------------------------
//*        CREATE FORTRAN SOURCE CODE
//GENFORT.FT05F001 DD *
userid.@.D5120.FORT77
4,5000
/*
//
```

Source list 5.6 JCL for the graph generation and vector source generation

5.4 Running on HITAC S820/80

In this section, we describe how to execute the integration and event generation jobs on a vector computer, i.e., HITAC S820/80.

Graph generation and source code generation

The source code generation program for the vector computer is installed, for the time being, only on FACOM at KEK. There are two different points between the JCLs' for the scalar and vector machines.

(1) `EXEC #GRACEV` will generate the source code for a vector processor.

(2) There are additional inputs, that is, the dimension (`NDIM`) and the number of calls (`NCALL`). These parameters will determine the vector length (See section 5.2).

Generation of library

In order to copy the generated file from FACOM to HITAC, user should

(1) convert a sequential file from the generated partitioned file on FACOM (See `'MLIB.UTILITY.CNTL(POTOPS)'`),

(2) get a file from FACOM using `ftp` on HITAC, and

(3) convert the sequential file to a partitioned file on HITAC.

User should edit a member `#GENLIB`.

```
//TI10HCL JOB CMD=NO,TIME=2,REGION=(4096K,5M),NOTIFY=TI10
//*     Mxxx.GRACE.Vyymmdd
//*MAIN PAGES=99999
//    EXEC F7E2HCL,
//        PARM.FORT='HAP,OPT(3),DCOM,COMARY,NOS,NOOPLIST,NAME',
//        PARM.LKED='LET,NCAL,EX=EA,LD=ANY'
//FORT.SYSLIB DD DSN=#KEKD.WWG.VECTOR.FORT,DISP=SHR
//FORT.SYSIN  DD DSN=#KEKD.WWG.VECTOR.FORT(AMPARM),DISP=SHR
//            DD DSN=#KEKD.WWG.VECTOR.FORT(AMPTBL),DISP=SHR
//            DD DSN=#KEKD.WWG.VECTOR.FORT(AMPSUM),DISP=SHR
//            DD DSN=#KEKD.WWG.VECTOR.FORT(AMPORD),DISP=SHR
```

Source list 5.7 JCL for making the library for amplitude calculation

continue to the next page

```
//             DD DSN=#KEKD.WWG.VECTOR.FORT(AM0001),DISP=SHR
//             DD DSN=#KEKD.WWG.VECTOR.FORT(AM0002),DISP=SHR
//             DD DSN=#KEKD.WWG.VECTOR.FORT(AM0003),DISP=SHR
//             DD DSN=#KEKD.WWG.VECTOR.FORT(AM0004),DISP=SHR
//             DD DSN=#KEKD.WWG.VECTOR.FORT(AM0005),DISP=SHR
//             DD DSN=#KEKD.WWG.VECTOR.FORT(AM0006),DISP=SHR
               .........................................
               .........................................
//             DD DSN=#KEKD.WWG.VECTOR.FORT(AM0023),DISP=SHR
//             DD DSN=#KEKD.WWG.VECTOR.FORT(AM0024),DISP=SHR
//             DD DSN=#KEKD.WWG.VECTOR.FORT(AM0025),DISP=SHR
//             DD DSN=#KEKD.WWG.VECTOR.FORT(AM0026),DISP=SHR
//             DD DSN=#KEKD.WWG.VECTOR.FORT(AM0027),DISP=SHR
//             DD DSN=#KEKD.WWG.VECTOR.FORT(AM0028),DISP=SHR
//*
//             DD DSN=#KEKD.GRACE.VECTOR.FORT(SMINIT),DISP=SHR
//             DD DSN=#KEKD.GRACE.VECTOR.FORT(SMEXTF),DISP=SHR
//             DD DSN=#KEKD.GRACE.VECTOR.FORT(SMEXTV),DISP=SHR
//             DD DSN=#KEKD.GRACE.VECTOR.FORT(SMPRPD),DISP=SHR
//             DD DSN=#KEKD.GRACE.VECTOR.FORT(SMINTF),DISP=SHR
//             DD DSN=#KEKD.GRACE.VECTOR.FORT(SMINTV),DISP=SHR
//             DD DSN=#KEKD.GRACE.VECTOR.FORT(SMCONF),DISP=SHR
//             DD DSN=#KEKD.GRACE.VECTOR.FORT(SMCONV),DISP=SHR
//             DD DSN=#KEKD.GRACE.VECTOR.FORT(SMCONS),DISP=SHR
//             DD DSN=#KEKD.GRACE.VECTOR.FORT(SMFFS),DISP=SHR
//             DD DSN=#KEKD.GRACE.VECTOR.FORT(SMFFV),DISP=SHR
//             DD DSN=#KEKD.GRACE.VECTOR.FORT(SMGGG),DISP=SHR
//             DD DSN=#KEKD.GRACE.VECTOR.FORT(SMGGGG),DISP=SHR
//             DD DSN=#KEKD.GRACE.VECTOR.FORT(SMSSS),DISP=SHR
//             DD DSN=#KEKD.GRACE.VECTOR.FORT(SMSSSS),DISP=SHR
//             DD DSN=#KEKD.GRACE.VECTOR.FORT(SMSSV),DISP=SHR
//             DD DSN=#KEKD.GRACE.VECTOR.FORT(SMSSVV),DISP=SHR
//             DD DSN=#KEKD.GRACE.VECTOR.FORT(SMVVV),DISP=SHR
//             DD DSN=#KEKD.GRACE.VECTOR.FORT(SMVVVV),DISP=SHR
//             DD DSN=#KEKD.GRACE.VECTOR.FORT(SMSVV),DISP=SHR
//*
//             DD DSN=#KEKD.GRACE.VECTOR.FORT(SPLTQ),DISP=SHR
//             DD DSN=#KEKD.GRACE.VECTOR.FORT(PHASEQ),DISP=SHR
//             DD DSN=#KEKD.GRACE.VECTOR.FORT(POLA),DISP=SHR
//             DD DSN=#KEKD.GRACE.VECTOR.FORT(FFS),DISP=SHR
//             DD DSN=#KEKD.GRACE.VECTOR.FORT(FFS0),DISP=SHR
//             DD DSN=#KEKD.GRACE.VECTOR.FORT(FFV),DISP=SHR
//             DD DSN=#KEKD.GRACE.VECTOR.FORT(FFV0),DISP=SHR
//             DD DSN=#KEKD.GRACE.VECTOR.FORT(SSV),DISP=SHR
//             DD DSN=#KEKD.GRACE.VECTOR.FORT(VVVV),DISP=SHR
//             DD DSN=#KEKD.GRACE.VECTOR.FORT(VVS),DISP=SHR
//LKED.SYSLMOD DD DSN=#KEKD.@.WWG.VECTOR.LOAD,
//            DISP=(RNW,CATLG,DELETE),
//            SPACE=(CYL,(2,1,50))
```

Source list 5.7 JCL for making the library for amplitude calculation

Notice:

In order to estimate the required memory size roughly, user may calculate the size of array "AG" which is included in common /AMGRPH/. As seen in member "INCL1", the size is NSIZE $\times$ LAG $\times$ NGRAPH. In our example of the process $e^+e^- \to W^+W^-\gamma$, these parameters are NSIZE $= 302$, LAG $= 72$ and NGRAPH $= 28$, then the size of this array becomes $16 \times 302 \times 72 \times 28 = 9741312$ bytes. In the real case, this program requires about 14 Mbytes.

When user changes the number of calls (NCALL), user should recreate the upper library. Otherwise the warning or fatal messages are seen.

Test of the generated source code

A member #TESTV, which is a JCL for a testing program, is automatically generated. This JCL is to execute the main program TESTV which is used to check the generated amplitudes at a point in the phase space. The user is recommended to confirm the gauge invariance and Lorentz frame independence before starting the integration by BASES.

```
//TI10TEST JOB CMD=NO,CLASS=I,REGION=(4096K,5M),NOTIFY=TI10
//*     MTAK.GRACE.Vyymmdd
//*MAIN PAGES=99999
//    EXEC F7E2HCLG,
//        PARM.FORT='HAP,OPT(3),DCOM,COMARY,NOOPLIST',
//        PARM.LKED='EX=EA,LD=ANY'
//FORT.SYSLIB DD DSN=#KEKD.WWG.VECTOR.FORT,DISP=SHR
//FORT.SYSIN  DD DSN=#KEKD.WWG.VECTOR.FORT(TESTV),DISP=SHR
//            DD DSN=#KEKD.WWG.VECTOR.FORT(USERIN),DISP=SHR
//            DD DSN=#KEKD.WWG.VECTOR.FORT(VBFNCT),DISP=SHR
//            DD DSN=#KEKD.WWG.VECTOR.FORT(USROUT),DISP=SHR
//*           DD DSN=#KEKD.WWG.VECTOR.FORT(SETMAS),DISP=SHR
//            DD DSN=#KEKD.WWG.VECTOR.FORT(KINIT),DISP=SHR
//            DD DSN=#KEKD.WWG.VECTOR.FORT(KINEM),DISP=SHR
//            DD DSN=#KEKD.GRACE.PROGS(BDUMMY),DISP=SHR
//LKED.SYSLIB DD
//            DD DSN=#KEKD.@.WWG.VECTOR.LOAD,DISP=SHR
//
```

Source list 5.8 JCL for the gauge invariance test

Numerical integration by the vector BASES

JCL for the numerical integration is generated in a member #INTV. By this JCL one can perform phase space integration with BASES. Don't forget to fix the filenames of kinematics.

```
//TI10WWA JOB CMD=NO,CLASS=I,REGION=(4096K,5M),NOTIFY=TI10
//*     MTAK.GRACE.Vyymmdd
//*MAIN PAGES=99999
//    EXEC F7E2HCLG,
//        PARM.FORT='HAP,OPT(3),DCOM,COMARY,NOS,NOOPLIST',
//        PARM.LKED='EX=EA,LD=ANY'
//FORT.SYSLIB DD DSN=#KEKD.WWG.VECTOR.FORT,DISP=SHR
//FORT.SYSIN  DD DSN=#KEKD.WWG.VECTOR.FORT(MAINVB),DISP=SHR
//            DD DSN=#KEKD.WWG.VECTOR.FORT(USERIN),DISP=SHR
//            DD DSN=#KEKD.WWG.VECTOR.FORT(VBFNCT),DISP=SHR
//            DD DSN=#KEKD.WWG.VECTOR.FORT(USROUT),DISP=SHR
//*           DD DSN=#KEKD.WWG.VECTOR.FORT(SETMAS),DISP=SHR
//            DD DSN=#KEKD.WWG.VECTOR.FORT(KINIT),DISP=SHR
//            DD DSN=#KEKD.WWG.VECTOR.FORT(KINEM),DISP=SHR
//            DD DSN=#KEKD.BASES50.VECTOR.FORT(DVFILL),DISP=SHR
//            DD DSN=#KEKD.BASES50.VECTOR.FORT(XVFILL),DISP=SHR
//            DD DSN=#KEKD.BASES50.VECTOR.FORT(VSHUPD),DISP=SHR
//LKED.SYSLIB DD
//            DD DSN=#KEKD.@.WWG.VECTOR.LOAD,DISP=SHR
//            DD DSN=#KEKD.BASES50.VECTOR.LOAD,DISP=SHR
//GO.SYSIN DD *
   1,1                 LOOP MIN AND MAX
  -4                   Print Flag
   0                   Flag
   9.9                 CPU TIME IN MINUITS
/*
//GO.FT23F001 DD DUMMY
//*GO.FT23F001 DD DSN=userid.PROCESS.DATA,DISP=(NEW,CATLG,DELETE),
//*          SPACE=(TRK,(5,5),RLSE),
//*          DCB=(RECFM=VBS,BLKSIZE=23476)
```

Source list 5.9 JCL for the numerical integration

Format of the output is described in section 3.5.6.
If user will execute the event generation, `userid.PROCESS.DATA` is required.

Event generation

JCL for the event generation on HITAC 820/20 is as follows:

```
//TI10SP JOB CMD=NO,CLASS=I,REGION=(4096K,5M),NOTIFY=TI10
//*     MTAK.GRACE.Vyymmdd
//*MAIN PAGES=99999
//    EXEC F7E2HCLG,
//        PARM.FORT='HAP,OPT(3),DCOM,COMARY,NOOPLIST',
//        PARM.LKED='EX=EA,LD=ANY'
```

Source list 5.10 JCL for the event generation

continue to the next page

```
//FORT.SYSLIB DD DSN=#KEKD.WWG.VECTOR.FORT,DISP=SHR
//FORT.SYSIN  DD DSN=#KEKD.WWG.VECTOR.FORT(MAINVS),DISP=SHR
//            DD DSN=#KEKD.WWG.VECTOR.FORT(USERIN),DISP=SHR
//            DD DSN=#KEKD.WWG.VECTOR.FORT(VBFNCT),DISP=SHR
//            DD DSN=#KEKD.WWG.VECTOR.FORT(USROUT),DISP=SHR
//*           DD DSN=#KEKD.WWG.VECTOR.FORT(SETMAS),DISP=SHR
//            DD DSN=#KEKD.WWG.VECTOR.FORT(KINIT),DISP=SHR
//            DD DSN=#KEKD.WWG.VECTOR.FORT(KINEM),DISP=SHR
//            DD DSN=#KEKD.WWG.VECTOR.FORT(SPINIT),DISP=SHR
//            DD DSN=#KEKD.WWG.VECTOR.FORT(SPEVNT),DISP=SHR
//            DD DSN=#KEKD.WWG.VECTOR.FORT(SPTERM),DISP=SHR
//            DD DSN=#KEKD.BASES50.VECTOR.FORT(DVFILL),DISP=SHR
//            DD DSN=#KEKD.BASES50.VECTOR.FORT(XVFILL),DISP=SHR
//            DD DSN=#KEKD.BASES50.VECTOR.FORT(VSHUPD),DISP=SHR
//LKED.SYSLIB DD
//            DD DSN=#KEKD.@.WWG.VECTOR.LOAD,DISP=SHR
//            DD DSN=#KEKD.BASES50.VECTOR.LOAD,DISP=SHR
//GO.SYSIN    DD *
 10000                 Number of events
   1.0                 CPU Time in Minutes
/*
//GO.FT23F001 DD DSN=userid.PROCESS.DATA,DISP=SHR,LABEL=(,,,IN)
```

Source list 5.10 JCL for the event generation

Chapter 6

Definition of the model

In chapter 2, we have described the Lagrangian of the standard model which our calculation based on. Here we summarize the Feynman rules of the models. These rules are given to `GRACE` through a model definition file. We also describe the structure and format of the file.

6.1 Feynman rules

6.1.1 Particles

We use names of particles as shown in table 6.1. There are two types of particle names in `GRACE`. Two character name is used to define external particles, to generate FORTRAN variables. This kind of names is defined in the model definition file, and these names can be changed. In order to identify special particles, predefined integer numbers(*particle id*) are assigned. `GRACE` recognizes through *particle id* which particle name corresponds to a special particle.

In the description of Feynman rules, ψ_n represents any fermion as a generic name. Similarly, ψ_I and ψ_i are generic names representing upper and lower component of a $SU(2)$ doublet fermion.

Throughout this section, fields in a vertex description represents incoming fields with incoming momenta to the vertex. That is, $\bar{\psi}_I \, \psi_i \, W^+_\alpha$ represents a vertex which corresponds to $< 0|T\mathcal{L}_{int} \, \bar{\psi}_I \, \psi_i \, W^+_\alpha|0 >$. The normalization of propagators and vertices are given in chapter 2.

field	*meaning*	*name in* GRACE	*particle id*
W_μ^+, W_μ^-	W boson	WB	2
Z_μ^0	Z boson	ZB	4
A_μ	photon	AB	1
G_μ	gluon	GL	8
χ^+, χ^-	charged Goldstone boson	XB	42
χ_3	neutral Goldstone boson	X3	41
ϕ	Higgs boson	PH	31
e^-	electron	EL	0
ν_e	electron neutrino	NE	0
μ^-	muon	MU	0
ν_μ	muon neutrino	NM	0
τ^-	tau lepton	TA	0
ν_τ	tau neutrino	NT	0
	any lepton	LU	0
	any neutrino	LD	0
u	u-quark	UQ	0
d	d-quark	DQ	0
s	s-quark	SQ	0
c	c-quark	CQ	0
b	b-quark	BQ	0
t	t-quark	TQ	0
	any up type quark	QU	0
	any down type quark	QD	0
c^A	ghost for A boson	CA	11
c^+	ghost for W boson	CP	12
c^-	ghost for W boson	CM	13
c^Z	ghost for Z boson	CZ	14
c^G	ghost for gluon	CG	18

Table 6.1 Particle names

6.1.2 Propagators

Propagators in electroweak theory are given by:

$$W_\mu^+ - - < - - W_\nu^- \quad \frac{1}{-k^2 - i\epsilon + M_W^2}[-g^{\mu\nu} + (1-\alpha_W)\frac{k^\mu k^\nu}{k^2 + i\epsilon - \alpha_W M_W^2}]$$

$$Z_\mu^+ - - < - - Z_\nu^- \quad \frac{1}{-k^2 - i\epsilon + M_Z^2}[-g^{\mu\nu} + (1-\alpha_Z)\frac{k^\mu k^\nu}{k^2 + i\epsilon - \alpha_Z M_Z^2}]$$

$$A_\mu - - < - - A_\nu \quad \frac{1}{-k^2 - i\epsilon}[-g^{\mu\nu} + (1-\alpha_A)\frac{k^\mu k^\nu}{k^2 + i\epsilon}]$$

$\chi^+ - - < - - \chi^-$	$\frac{1}{-k^2 - i\epsilon + \alpha_W M_W^2}$
$\chi_3 - - < - - \chi_3$	$\frac{1}{-k^2 - i\epsilon + \alpha_Z M_Z^2}$
$\phi - - < - - \phi$	$\frac{1}{-k^2 - i\epsilon + m_H^2}$
$\psi - - < - - \bar{\psi}$	$\frac{m_f + \not{k}}{-k^2 - i\epsilon + m_f^2}$
$c^+ - - < - - \bar{c}^-$	$\frac{1}{-k^2 - i\epsilon + \alpha_W M_W^2}$
$c^- - - < - - \bar{c}^+$	$\frac{1}{-k^2 - i\epsilon + \alpha_W M_W^2}$
$c^Z - - < - - \bar{c}^Z$	$\frac{1}{-k^2 - i\epsilon + \alpha_Z M_Z^2}$
$c^A - - < - - \bar{c}^A$	$\frac{1}{-k^2 - i\epsilon}$
$G_\mu - - < - - G_\nu$	$\frac{1}{-k^2 - i\epsilon}[-g^{\mu\nu} + (1 - \alpha_G)\frac{k^\mu k^\nu}{k^2 + i\epsilon}]$
$c^G - - < - - \bar{c}^G$	$\frac{1}{-k^2 - i\epsilon}$

6.1.3 Vector-vector-vector vertex

Vertex of three vector bosons in electroweak theory takes the following form for vector fields $V_\alpha^1(k), V_\beta^2(p)$ and $V_\gamma^3(q)$:

$$\mathtt{CVVV}[(k-p)_\gamma g_{\alpha\beta} + (p-q)_\alpha g_{\beta\gamma} + (q-k)_\beta g_{\gamma\alpha}]$$

where, coupling constant `CVVV` is given by:

vertex	CVVV
$W_\alpha^-(k)\, W_\beta^+(p)\, Z_\gamma(q)$	$\mathtt{CZWW} = \frac{eM_W}{\sqrt{M_Z^2 - M_W^2}}$
$W_\alpha^-(k)\, W_\beta^+(p)\, A_\gamma(q)$	$\mathtt{CAWW} = e$

Three gluon vertex in QCD for $G_\alpha^a(k), G_\beta^b(p)$ and $G_\gamma^c(q)$:

$$\mathtt{CQCD}\,(-if^{abc})[(k-p)_\gamma g_{\alpha\beta} + (p-q)_\alpha g_{\beta\gamma} + (q-k)_\beta g_{\gamma\alpha}]$$

6.1.4 Vector-vector-vector-vector vertex

Vertex of four vector bosons in electroweak takes the following form for vector fields $V^1_\alpha(k)$, $V^2_\beta(p)$, $V^3_\gamma(q)$ and $V^4_\delta(r)$:

$$\mathtt{CVVVV}[g_{\alpha\gamma}g_{\beta\delta} + g_{\alpha\delta}g_{\beta\gamma} - 2g_{\alpha\beta}g_{\gamma\delta}]$$

where, coupling constant CVVVV is:

vertex	CVVVV
$W^-_\alpha\, W^+_\beta\, A_\gamma\, A_\delta$	$\mathtt{CWWAA} = e^2$
$W^-_\alpha\, W^+_\beta\, Z_\gamma\, A_\delta$	$\mathtt{CWWZA} = \dfrac{e^2 M_W}{\sqrt{M_Z^2 - M_W^2}}$
$W^-_\alpha\, W^+_\beta\, Z_\gamma\, Z_\delta$	$\mathtt{CWWZZ} = \dfrac{e^2 M_W^2}{M_Z^2 - M_W^2}$
$W^-_\alpha\, W^-_\beta\, W^+_\gamma\, W^+_\delta$	$\mathtt{CWWWW} = \dfrac{-\,e^2 M_Z^2}{M_Z^2 - M_W^2}$

Four gluon vertex in QCD for $G^a_\alpha(k), G^b_\beta(p),\ G^c_\gamma(q) and G^d_\delta(r)$ is given by:

$$-\mathtt{CQCD}^2 \quad [(f^{ace}f^{bde} - f^{ade}f^{cbe})g_{\alpha\beta}g_{\gamma\delta} + (f^{abe}f^{cde} - f^{ade}f^{bce})g_{\alpha\gamma}g_{\beta\delta} \\ +(f^{ace}f^{dbe} - f^{abe}f^{cde})g_{\alpha\delta}g_{\gamma\beta}]$$

6.1.5 Fermion-fermion-vector vertex

Vertex of two fermions and a vector in electroweak takes the following form for fermion fields ψ, $\bar{\psi}$ and vector field V_α:

$$\mathtt{CVFF(1)}\gamma_\alpha \frac{1-\gamma_5}{2} + \mathtt{CVFF(2)}\gamma_\alpha \frac{1+\gamma_5}{2}$$

where, coupling constant CVFF is given by (neglecting mixing matrix):

vertex	CVFF
$\bar{\psi}_I\, \psi_i\, W^+_\alpha$	$\begin{cases} \mathtt{CWFF(1,1)} = \dfrac{eM_Z}{\sqrt{2(M_Z^2 - M_W^2)}} U^\dagger_{Ii} \\ \mathtt{CWFF(2,1)} = 0 \end{cases}$

$\bar{\psi}_i\,\psi_I\,W^-_\alpha$	$\begin{cases}\texttt{CWFF(1,2)} = \texttt{CONJG(CWFF(1,1))}\\ \texttt{CWFF(2,2)} = \texttt{CONJG(CWFF(2,1))}\end{cases}$
$\bar{\psi}_n\,\psi_n\,A_\alpha$	$\begin{cases}\texttt{CAFF(1)} = eQ_n\\ \texttt{CAFF(2)} = eQ_n\end{cases}$
$\bar{\psi}_I\,\psi_I\,Z_\alpha$	$\begin{cases}\texttt{CZFU(1)} = \dfrac{eM_Z^2}{2M_W\sqrt{M_Z^2-M_W^2}}(1-2Q_I\dfrac{M_Z^2-M_W^2}{M_Z^2})\\ \texttt{CZFU(2)} = \dfrac{eM_Z^2}{2M_W\sqrt{M_Z^2-M_W^2}}(-2Q_I\dfrac{M_Z^2-M_W^2}{M_Z^2})\end{cases}$
$\bar{\psi}_i\,\psi_i\,Z_\alpha$	$\begin{cases}\texttt{CZFD(1)} = \dfrac{eM_Z^2}{2M_W\sqrt{M_Z^2-M_W^2}}(-1-2Q_i\dfrac{M_Z^2-M_W^2}{M_Z^2})\\ \texttt{CZFD(2)} = \dfrac{eM_Z^2}{2M_W\sqrt{M_Z^2-M_W^2}}(-2Q_i\dfrac{M_Z^2-M_W^2}{M_Z^2})\end{cases}$

When mixing among quarks are incorporated, more vertices appear into to model. Fermion-gluon vertex is given by:

$$\texttt{CQCD}\,T^a_{ij}\,\gamma_\mu$$

6.1.6 Scalar-scalar-vector vertex

Vertex of two scalars and a vector in electroweak takes the following form for scalar fields $S^1(p), S^2(q)$ and vector a field V^3_α:

$$\texttt{CVSS}(p-q)_\alpha$$

where, coupling constant CVSS is given by:

vertex	CVSS
$\chi^-(p)\,\phi(q)\,W^+_\alpha$	$\texttt{CWXP(1)} = \dfrac{ieM_Z}{2\sqrt{M_Z^2-M_W^2}}$
$\chi^+(p)\,\phi(q)\,W^-_\alpha$	$\texttt{CWXP(2)} = -\texttt{CONJG(CWXP(1))}$
$\chi^-(p)\,\chi_3(q)\,W^+_\alpha$	$\texttt{CWX3(1)} = \dfrac{eM_Z}{2\sqrt{M_Z^2-M_W^2}}$

$\chi^+(p)\,\chi_3(q)\,W^-_\alpha$	$\texttt{CWX3(2)} = -\texttt{CONJG(CWX3(1))}$
$\chi^-(p)\,\chi^+(q)\,Z_\alpha$	$\texttt{CZXX} = \dfrac{e(M_Z^2 - 2M_W^2)}{2M_W\sqrt{M_Z^2 - M_W^2}}$
$\chi^-(p)\,\chi^+(q)\,A_\alpha$	$\texttt{CAXX} = -e$
$\chi_3(p)\,\phi(q)\,Z_\alpha$	$\texttt{CZ3P} = \dfrac{ieM_Z^2}{2M_W\sqrt{M_Z^2 - M_W^2}}$

6.1.7 Scalar-vector-vector vertex

Vertex of a scalar and two vectors in electroweak takes the following form for a scalar field S, and vector fields V_α, V_β:

$$\texttt{CSVV}g_{\alpha\beta}$$

where, coupling constant CSVV is given by:

vertex	CSVV
$\phi\,W^-_\alpha\,W^+_\beta$	$\texttt{CPWW} = \dfrac{eM_WM_Z}{\sqrt{M_Z^2 - M_W^2}}$
$\phi\,Z_\alpha\,Z_\beta$	$\texttt{CPZZ} = \dfrac{eM_Z^3}{M_W\sqrt{M_Z^2 - M_W^2}}$
$\chi^-\,W^+_\alpha\,Z_\beta$	$\texttt{CXWZ(1)} = ie\sqrt{M_Z^2 - M_W^2}$
$\chi^+\,W^-_\alpha\,Z_\beta$	$\texttt{CXWZ(2)} = \texttt{CONJG(CXWZ(1))}$
$\chi^-\,W^+_\alpha\,A_\beta$	$\texttt{CXWA(1)} = -ieM_W$
$\chi^+\,W^-_\alpha\,A_\beta$	$\texttt{CXWA(2)} = \texttt{CONJG(CXWA(1))}$

6.1.8 Scalar-scalar-vector-vector vertex

Vertex of two scalars and two vectors in electroweak theory takes the following form for scalar fields S, S and vector fields V_α, V_β:

$$\mathtt{CSSVV}\, g_{\alpha\beta}$$

where, coupling constant `CSSVV` is given by:

vertex	CSSVV
$\phi\,\phi\,W^-_\alpha\,W^+_\beta$	$\mathtt{CPPWW} = \dfrac{e^2 M_Z^2}{2(M_Z^2 - M_W^2)}$
$\phi\,\phi\,Z_\alpha\,Z_\beta$	$\mathtt{CPPZZ} = \dfrac{e^2 M_Z^4}{2M_W^2(M_Z^2 - M_W^2)}$
$\phi\,\chi^-\,W^+_\alpha\,Z_\beta$	$\mathtt{CPXWZ(1)} = \dfrac{ie^2 M_Z}{2M_W}$
$\phi\,\chi^+\,W^-_\alpha\,Z_\beta$	$\mathtt{CPXWZ(2)} = \mathtt{CONJG(CPXWZ(1))}$
$\phi\,\chi^-\,W^+_\alpha\,A_\beta$	$\mathtt{CPXWA(1)} = \dfrac{-\,ie^2 M_Z}{2\sqrt{M_Z^2 - M_W^2}}$
$\phi\,\chi^+\,W^-_\alpha\,A_\beta$	$\mathtt{CPXWA(2)} = \mathtt{CONJG(CPXWA(1))}$
$\chi_3\,\chi_3\,W^-_\alpha\,W^+_\beta$	$\mathtt{C33WW} = \dfrac{e^2 M_Z^2}{2(M_Z^2 - M_W^2)}$
$\chi_3\,\chi_3\,Z_\alpha\,Z_\beta$	$\mathtt{C33ZZ} = \dfrac{e^2 M_Z^4}{2M_W^2(M_Z^2 - M_W^2)}$
$\chi_3\,\chi^-\,W^+_\alpha\,Z_\beta$	$\mathtt{C3XWZ(1)} = \dfrac{e^2 M_Z}{2M_W}$
$\chi_3\,\chi^+\,W^-_\alpha\,Z_\beta$	$\mathtt{C3XWZ(2)} = \mathtt{CONJG(C3XWZ(1))}$
$\chi_3\,\chi^-\,W^+_\alpha\,A_\beta$	$\mathtt{C3XWA(1)} = \dfrac{-\,e^2 M_Z}{2\sqrt{M_Z^2 - M_W^2}}$
$\chi_3\,\chi^+\,W^-_\alpha\,A_\beta$	$\mathtt{C3XWA(2)} = \mathtt{CONJG(C3XWA(1))}$

$\chi^-\,\chi^+\,W^-_\alpha\,W^+_\beta$	`CXXWW` $= \dfrac{e^2 M_Z^2}{2(M_Z^2 - M_W^2)}$
$\chi^-\,\chi^+\,Z_\alpha\,Z_\beta$	`CXXZZ` $= \dfrac{e^2(2M_W^2 - M_Z^2)^2}{2M_W^2(M_Z^2 - M_W^2)}$
$\chi^-\,\chi^+\,A_\alpha\,A_\beta$	`CXXAA` $= 2e^2$
$\chi^-\,\chi^+\,A_\alpha\,Z_\beta$	`CXXAZ` $= \dfrac{e^2(2M_W^2 - M_Z^2)}{M_W\sqrt{M_Z^2 - M_W^2}}$

6.1.9 Scalar-scalar-scalar vertex

Vertex of three scalars in electroweak theory is constant factor:

`CSSS`

where, `CSSS` is given by:

vertex	`CSSS`
$\phi\,\chi^-\,\chi^+$	`CPXX` $= \dfrac{-\,e m_H^2 M_Z}{2M_W\sqrt{M_Z^2 - M_W^2}}$
$\phi\,\chi_3\,\chi_3$	`CP33` $= \dfrac{-\,e m_H^2 M_Z}{2M_W\sqrt{M_Z^2 - M_W^2}}$
$\phi\,\phi\,\phi$	`CPPP` $= \dfrac{-\,3e m_H^2 M_Z}{2M_W\sqrt{M_Z^2 - M_W^2}}$

6.1.10 Scalar-scalar-scalar-scalar vertex

Vertex of four scalars in electroweak theory is constant factor:

`CSSSS`

where, coupling constant `CSSSS` is given by:

vertex	`CSSSS`
$\chi_3\,\chi_3\,\chi_3\,\chi_3$	`C3333` $= \dfrac{-\,3e^2 m_H^2 M_Z^2}{4M_W^2(M_Z^2 - M_W^2)}$

$\phi\,\phi\,\phi\,\phi$	$\texttt{CPPPP} = \dfrac{-\,3e^2 m_H^2 M_Z^2}{4M_W^2(M_Z^2 - M_W^2)}$
$\chi^-\,\chi^+\,\chi_3\,\chi_3$	$\texttt{CXX33} = \dfrac{-\,e^2 m_H^2 M_Z^2}{4M_W^2(M_Z^2 - M_W^2)}$
$\chi^-\,\chi^+\,\phi\,\phi$	$\texttt{CXXPP} = \dfrac{-\,e^2 m_H^2 M_Z^2}{4M_W^2(M_Z^2 - M_W^2)}$
$\phi\,\phi\,\chi_3\,\chi_3$	$\texttt{CPP33} = \dfrac{-\,e^2 m_H^2 M_Z^2}{4M_W^2(M_Z^2 - M_W^2)}$
$\chi^-\,\chi^-\,\chi^+\,\chi^+$	$\texttt{CXXXX} = \dfrac{-\,e^2 m_H^2 M_Z^2}{2M_W^2(M_Z^2 - M_W^2)}$

6.1.11 Fermion-fermion-scalar vertex

Vertex of two fermions and a vector boson in electroweak theory takes the following form for fermion fields ψ, $\bar{\psi}$ and scalar field S:

$$\texttt{CSFF(1)}\frac{1-\gamma_5}{2} + \texttt{CSFF(2)}\frac{1+\gamma_5}{2}$$

where, coupling constant CSFF is given by:

vertex	CSFF
$\bar{\psi}_I\,\psi_i\,\chi^+$	$\begin{cases} \texttt{CXFF(1,1)} = \dfrac{iem_I M_Z}{M_W\sqrt{2(M_Z^2 - M_W^2)}} U^\dagger_{Ii} \\ \texttt{CXFF(2,1)} = \dfrac{-\,iem_i M_Z}{M_W\sqrt{2(M_Z^2 - M_W^2)}} U^\dagger_{Ii} \end{cases}$
$\bar{\psi}_i\,\psi_I\,\chi^-$	$\begin{cases} \texttt{CXFF(1,2)} = \texttt{CONJG(CXFF(2,1))} \\ \texttt{CXFF(2,2)} = \texttt{CONJG(CXFF(1,1))} \end{cases}$
$\bar{\psi}_n\,\psi_n\,\phi$	$\begin{cases} \texttt{CPFF(1)} = \dfrac{-\,em_n M_Z}{2M_W\sqrt{M_Z^2 - M_W^2}} \\ \texttt{CPFF(2)} = \dfrac{-\,em_n M_Z}{2M_W\sqrt{M_Z^2 - M_W^2}} \end{cases}$

$\bar{\psi}_I \, \psi_I \, \chi_3$
$$\begin{cases} \texttt{C3FU(1)} = \dfrac{iem_I M_Z}{2M_W\sqrt{M_Z^2 - M_W^2}} \\ \texttt{C3FU(2)} = \dfrac{-\,iem_I M_Z}{2M_W\sqrt{M_Z^2 - M_W^2}} \end{cases}$$

$\bar{\psi}_i \, \psi_i \, \chi_3$
$$\begin{cases} \texttt{C3FU(1)} = \dfrac{-\,iem_i M_Z}{2M_W\sqrt{M_Z^2 - M_W^2}} \\ \texttt{C3FU(2)} = \dfrac{iem_i M_Z}{2M_W\sqrt{M_Z^2 - M_W^2}} \end{cases}$$

6.2 File format of model definition

The default models used in `GRACE` are defined in the file

“`$(GRACEDIR)/data/particle.table`”.

This table includes QED, electroweak and QCD Feynman rules described in the previous section. Although this table does not include quark mixing, extention is straight forward for the case with quark mixing.

We provide another model definition file

“`$(GRACEDIR)/data/particle.table0`”.

In this table the interactions are omitted, which give rise to very small contributions proportional to m_e^2. Particularly, this may give negative cross section for polarized process. For example, $e^+e^-\chi_3$ vertex is omitted since whose coupling constant is proportional to m_e^2.

We show the structure of this kind of file in Fig.6.1.
The meaning of each line is defined by the first letter in the line.

(1) When the first letter is “`*`”, then the line is comment line.
The first line of a file is used as the title line to distinguish this file from others. Thus the first line of a file must be a comment line.

(2) The file is composed of the following three parts:

1) definition of particles

2) definition of interaction vertices

3) definition of default values of masses and widths of particles.

4) definition of default values of coupling constants.

The line which begins with the letter “`E`” is used to specify the end of each part. The rest of the line should be blanks.

```
*PTCLTBL  Electro-Weak and QCD, no Cabbibo mixing, with Scalar
*******************************************************************
* Particle Block            <---- definition of particles
*
* Particles  Fermion Charge Spin*2    Type Color
  WB           0       1     2         2    1     W Boson
  ZB           0       0     2         4    1     Z Boson
                 ...
E                           <---- end of definition
*******************************************************************
* Vertex Block              <---- definition of vertex
*
* LEGS  VERTEX TABLE    EORD    WORD    CORD   NAME
* Gauge-boson Three-vertices
   3   ZB WB WB          0       1       0    CZWW
   3   AB WB WB          0       1       0    CAWW
                 ...
   3   WB NE EL          0       1       0    CWEL(2,2)
   3   WB NM MU          0       1       0    CWMU(2,2)
                 ...
E                           <---- end of definition

*******************************************************************
* Mass                      <---- set the constant parameters
      AMWB = 80.0D0
      AMZB = 91.1D0
                 ...         particle masses
*
* Width
      AGWB = 0.0D0
      AGZB = 0.0D0
                 ...         total decay width of particle
E
* Coupling constants
C------------------------------------------------------------------
      AMWB2 = AMWB*AMWB
      AMZB2 = AMZB*AMZB
                 ...         set the parameters used at the vertex
* VVV
      CZWW    =  CE*GW
      CAWW    =  CE
                 ...
      CWL (1,1) = GWFL
      CWL (2,1) = GWFR
      CWL (1,2) = CONJG(CWL (1,1))
      CWL (2,2) = CONJG(CWL (2,1))
      CWEL(1,1) = GWFL
                 ...         definition of coupling
*
```

Fig. 6.1 Structure of Particle table

(3) Other lines are used to describe properties of particles or vertices. These lines should begin with `blank` character.

In the following subsections, we describe the specification of the first two parts of the file, i.e., definition of particles and interaction vertices.

The last two parts are used to define default values of parameters, such as value of particles masses or value of coupling constants. Since these parameters are not independent, and many of them are calculated from independent parameters. These parts are written in the form of FORTRAN code, by which parameters are calculated. Actually in the generated code, third and fourth part is copied to the subroutines `SETMAS` and `AMPARM`, respectively (See section 3.2). Subroutine `USERIN` calls `SETMAS` and `AMPARM` in this order at the beginning of the calculation of amplitudes. Default values of masses and widths are defined in the subroutine `SETMAS` and value of coupling constants are calculated in the subroutine `AMPARM` by using parameters defined in `SETMAS`. One can change the values of masses and widths before calling `AMPARM` in the subroutine `USERIN`.

6.2.1 Definition of particles

Particles which participate into the process are defined in this table. The general format is as follows;

1) One line is reserved for one particle.

2) The properties of the particle should be written from the second column and each property must be separated by at least one blank.

The items of particle property are described below (item number is counted from left to right on a line).

1) **Name of particles**
The name should be two characters (one character is not allowed). In the generated FORTRAN code, the mass of the particle is expressed by attaching "`AM`" in front of the name and the total decay width is given by the prefix "`AG`".

2) **Fermion Number**
If the particle is not fermion then 0. Otherwise it is positive integer, which is common to fermions with same conserving fermion number.

3) **Electric charge**
Assign "$(charge) - 2 \times (baryon\ number)$", where `charge` is the electric charge in the unit of positron charge e.

4) **twice of spin**
Assign twice of the intrinsic spin of the particle.

5) **particle-id**
In order to specify special particles, identifying number is assigned to these particles as shown in the above table. To other particles, 0 is assigned.

6) **Color**
The QCD color dimension of particles (singlet:1, triplet:3, octet:8, ...).

After the sixth item, characters are treated as comments.

6.2.2 Definition of vertices

The general format is as follows;

1) One line is reserved for one kind of vertex.

2) The properties of the vertex should be written from the second column and each property must be separated by at least one blank.

The items of vertex property are described below (item number is counted from left to right on a line).

1) number of particles connected with the vertex.

2) Names of particles connected with the vertex.
List the names of all the particles, defined in the definition of particles, with the spacing at least one blank. When the same kind of particles appear more than once, repeat the name same times of its appearance.

3) The order of QED interaction(`EORDER`).
This is equal to the power of the coupling constant e in the given process.

4) The order of electroweak interaction (`WORDER`).
In our definition of interactions of electroweak theory, all the couplings contains the charge e. Thus this `WORDER` is equal to the power of e in the process.

5) The order of QCD interaction(`CORDER`). It is allowed to give this item together with `EORDER` and/or `WORDER` specification.

6) The name of the variable corresponding to coupling constant used in the generated FORTRAN code (within 6 characters).
The name defined here is regarded as complex variable in the generated FORTRAN source code.
A variable may be an array. There are left- and right handed coupling constants in fermion-boson vertices. These two constants are stored in a array with two components as an example of $\bar{\psi}_i\psi_i Z_\alpha$ vertex shown in the previous section. When coupling constant of a vertex is different from one of its charge conjugated vertex, as $\xi^{\pm}\phi W^{\mp}$ vertex, it is stored as an element of array with its charge conjugated value.

Chapter 7

Libraries for the amplitude calculation

These libraries are developed for calculation of scattering amplitudes. `GRACE` generates FORTRAN source code which call subroutines in these libraries.

First we fix our notations.

1. Feynman rules
   Feynman rules are given in chapter 2 and 6.

2. Four momentum
   Momentum is expressed by `REAL*8 P(4)` whose 4-th component is the energy of the particle.

3. Spinor
   The spinor is normalized as

$$u(p,h)\overline{u}(p,h) = (\not{p} + m)\frac{1 + h\gamma_5\not{s}}{2},$$

$$v(p,h)\overline{v}(p,h) = (\not{p} - m)\frac{1 + h\gamma_5\not{s}}{2},$$

   where helicity $h = \pm 1$ and spacial components of spin vector s is proportional to three momentum of the particle, since we consider helicity states.

4. Polarization vector
   The normalization of the polarization vector of vector particle, follows the definition given in chapter 2:

$$\sum_{spin} \epsilon_\mu \epsilon_\nu = -g_{\mu\nu} + \frac{q_\mu q_\nu}{M^2}, \qquad \text{etc.}$$

5. Spin component
   The components of spin and polarization vector are specified by the following numbering of the index,

Fermion	: 0 (helicity -1), 1 (helicity $+1$)
Vector boson	: 0, 1 (transverse), 2 (longitudinal)

6. Gauge parameters
   By generated FORTRAN code, one can calculate the amplitude in general covariant gauge. The gauge parameter which appears in both of polarization vector and the denominator of propagator of vector boson is defined by the array

   ```
   COMMON /SMGAUG/AGAUGE(0:4)
   REAL*8 AGAUGE.
   ```

   The element of this array is controlled by the integer variable

   ```
   COMMON /SMGAUS/IGAU00, IGAUAB, IGAUWB, IGAUZB, IGAUGL.
   ```

   Each of variables `IGAUAB, IGAUWB, IGAUZB, IGAUGL` takes `1, 2, 3, 4`, corresponding to, $\gamma, W^{\pm}, Z^0$ and *gluon*, respectively. The values of gauge parameters are stored in `AGAUGE(1) ...  AGAUGE(4)`. The default values are given by subroutine `AMPARM`. If one wants to use unitary gauge, set corresponding variables of `IGAUAB, IGAUWB, IGAUZB, IGAUGL` to 0. In this case irrelevant graphs are automatically dropped.

7.1 Generated FORTRAN source code

We describe an outline of method of calculation of amplitude by generated code (see also section 3.2).

1. At the first stage the external lines are processed.
   The subroutine `SMEXTF` is called for each fermion and `SMEXTV` for vector particle. They construct tables of information about external lines necessary for the succeeding calculation.

2. Define internal momenta.
   Internal momenta are defined as linear combinations of external momenta and denominator of the propagators are calculated.

3. Next internal lines are processed.
   In the method of `CHANEL`, numerator of propagators are decomposed as bi-linear form of on-shell wave functions. For this purpose, the subroutine `SMINTF` is called for the numerator of the propagator of each fermion and `SMINTV` for that of vector particle.

4. Then the vertex is calculated.
   According to the kind of vertex, the following subroutine is called:

`SMFFV`	Fermion-fermion-vector
`SMFFS`	Fermion-fermion-scalar
`SMVVV`	Vector-vector-vector
`SMSVV`	Scalar-vector-vector
`SMSSV`	Scalar-scalar-vector
`SMSSS`	Scalar-scalar-scalar
`SMVVVV`	Vector-vector-vector-vector
`SMSSVV`	Scalar-scalar-vector-vector
`SMSSSS`	Scalar-scalar-scalar-scalar
`SMGGG`	Gluon-gluon-gluon
`SMGGGG`	Gluon-gluon-gluon-gluon

   Values of coupling constants used by them are given in the chapter 6. The result of calculation of vertex amplitude is stored in a table for each helicity states. This table is composed of two arrays `LT` (information about the size this table) and `AV` (values of amplitude).

5. Connect the vertices by internal lines.
   Since vertices are connected by propagators in the Feynman graph, table of amplitudes for vertices are connected by summing over indices introduced when propagators are decomposed. The following subroutines are called to connecting tables of amplitudes.

`SMCONF`	connect by a fermion propagator
`SMCONV`	connect by a vector propagator
`SMCONS`	connect by a scalar propagator

6. Ordering of particles in the table of amplitude
   Intermediate results appear as connected tables of vertex amplitudes. They are connected again and again until amplitude of whole Feynman graph is obtained. Since propagators to be connected is specified by the position in the table, knowledge about the ordering convention of particles in the table is necessary to check the validity of generated source code.

   When an amplitude of a vertex is calculated by calling the corresponding subroutine, the ordering of particles in the table is represented by the name of the subroutine and ordering of arguments of the subroutine called. For example, in the subroutine `SMFFV` the amplitude table is arranged in the order of *F(fermion), F(fermion) and V(vector).* Ordering of two fermions are determined by the ordering of the arguments. The exact description of ordering in the vertex calculation is given in the description of each subroutine.

   When two particles are taken from two tables and connected, the resultant ordering of particles in the table is obtained by merging two sequence of ordered particles omitting the connected particles. For example, suppose one has tables of amplitudes (particle-1, particle-2, particle-3) and (particle-4, particle-3, particle-5). Then the particle-3 can be connected and the resultant amplitude has a table (particle-1, particle-2, particle-4, particle-5). The final form of the table corresponds to the Feynman graph. However it is possible that the ordering of the external particles in the tables are different graph to graph. Before amplitudes are summed over graphs, standardization of the ordering is necessary. For this purpose subroutine AMPORD is applied to the resultant table of amplitude for each graph.

7. After summing over all diagrams, the helicity amplitudes is squared. If necessary, spin states are summed further. It is done by subroutine AMPSUM, which is included in the generated code.

7.2 Interface routines to CHANEL

7.2.1 External particle

Before calculation of vertex amplitude, tables of information about external and internal particles are prepared.

External fermion

External fermion line has its own table used for calculating vertex amplitude. It is obtained by:

```
CALL SMEXTF(IO, AM, PE, PS, CE)

INTEGER        IO              input
REAL*8         AM              input
REAL*8         PE(4)           input
REAL*8         PS(4,2)         output
COMPLEX*16     CE(2,2)         output
```

1. The variable IO takes the value 1, if the input spinor is u or v, and 2, if $\overline{u}$ or $\overline{v}$.

2. AM means the mass of fermion.

3. PE is an array of four-momentum.

4. The output variables PS, CE are used in SMFFV and SMFFS.

5. At the same time another variable used in SMFFV, SMFFS and SMCOMF.

```
REAL*8         EW(1)
EW(1) = (1 for particle, -1 for anti-particle )
```

is also defined in the generated code.

External vector boson

External vector particles also have corresponding tables:

```
CALL SMEXTV(LP, AM, PE, EP, EW, IGAUG)

INTEGER        LP             input
REAL*8         AM             input
REAL*8         PE(4)          input
REAL*8         EP(4,LP)       output
REAL*8         EW(LP)         output
INTEGER        IGAUG          input
```

1. `LP` is the freedom of polarization vector.
   It is 2 for A_μ, G_μ and 3 for $W_\mu^\pm$, Z_μ.
   Here A_μ, G_μ, $W_\mu^\pm$ and Z_μ represent photon, gluon, $W^\pm$ and Z^0 bosons.

2. `AM` is the mass of particle.

3. `PE` is an array of four-momentum.

4. The output variable `EP` is the table of the polarization vectors. The first index indicates the components of four-vector and the second index classifies 1, 2 : transverse, 3 : longitudinal polarization vector.

5. The output variable `EW` is used in `SMCONV`.

6. `IGAUG` is used to select the gauge.
   If it is 0, then unitary gauge, otherwise general covariant gauge. The value of gauge parameter is taken from a component `AGAUGE(IGAUG)` of array defined as `REAL*8 AGAUGE(0:4)` in the common block `/SMGAUG/`.

7.2.2 Numerator of propagator

Fermion propagator

The following subroutine calculates a table for numerator of fermion propagator.

```
CALL SMINTF(AM, PE, VM, EW, PS, CE)

REAL*8          AM              input
REAL*8          PE(4)           input
REAL*8          VM              input
REAL*8          EW(2)           output
REAL*8          PS(4,3)         output
COMPLEX*16      CE(2,4)         output
```

1. AM is the mass of particle.
2. PE is an array of four-momentum.
3. VM is the square of four-momentum. Sometimes numerical cancellation appears in the direct calculation of this quantity by PE(0)**2 - PE(1)**2 - PE(2)**2 - PE(3)**2. In the generated code, this is calculated from inner products between external momenta.
4. EW is used in SMFFV, SMFFS and SMCOMF.
5. The output variables PS, CE are used in SMFFV and SMFFS.

Numerator of vector boson propagator

Internal vector particles also have corresponding tables:

```
CALL SMINTV(LP, AM, PE, EP, EW, VM, IGAUG)

INTEGER         LP              input
REAL*8          AM              input
REAL*8          PE(4)           input
REAL*8          EP(4,LP)        output
REAL*8          EW(LP)          output
REAL*8          VM              input
INTEGER         IGAUG           input
```

1. LP is the freedom of polarization vector.
   It is 3 for A_μ, G_μ and 4 for $W^\pm_\mu$, Z_μ.
   Here A_μ, G_μ, $W^\pm_\mu$ and Z_μ represent photon, gluon, $W^\pm$ and Z^0 bosons.
2. AM is the mass of particle.
3. PE is an array of four-momentum.

4. The output variable EP is the table of the polarization vectors. The first index indicates the components of four-vector and the second index classifies 1, 2 : transverse, 3 : longitudinal, 4 : virtual polarization vector.

5. The output variable EW is used in SMCOMV.

6. VM is the square of four-momentum. Sometimes numerical cancellation appears in the direct calculation of this quantity by `PE(0)**2 - PE(1)**2 - PE(2)**2 - PE(3)**2`. In the generated code, this is calculated from inner products between external momenta.

7. IGAUG is used to select the gauge.
   If it is 0, then unitary gauge, otherwise general covariant gauge. The value of gauge parameter is taken from a component AGAUGE(IGAUG) of array defined as `REAL*8 AGAUGE(0:4)` in the common block /SMGAUG/.

7.2.3 Denominator of propagator

This subroutine calculates the denominator of the propagator.

```
      CALL SMPRPD(APROP, AMOMQ, AMASSQ, AMAG)

      COMPLEX*16      APROP           input/output
      REAL*8          AMOMQ           input
      REAL*8          AMASSQ          input
      REAL*8          AMAG            input
```

1. APROP is the product of denominators of propagators. This subroutines calculates

   APROP = (denominator of a propagator)* APROP.

2. AMOMQ is the square of four-momentum. Sometimes numerical cancellation appears in the direct calculation of this quantity by `P(0)**2 - P(1)**2 - P(2)**2 - P(3)**2`. In the generated code, this is calculated from inner products between external momenta.

3. AMASSQ is the square of mass of the particle.

4. AMAG is the product of mass and width of the particle.

7.2.4 Vertices

Fermion-fermion-vector coupling

This calculates the fermion-fermion-vector vertex

$$\overline{U}_2(p_2)\not{\epsilon}\left(C_L\frac{1-\gamma_5}{2}+C_R\frac{1+\gamma_5}{2}\right)U_1(p_1),$$

where U represents a spinor u or v. It returns the results for all combinations of helicity and polarization states (See section 6.1.5).

```
      CALL SMFFV(L1, L2, LV, EW1, EW2, AM1, AM2, CPL, CE1, CE2,
     &           PS1, PS2, EP, LT, AV)

      INTEGER        L1, L2             input
      INTEGER        LV                 input
      REAL*8         EW1(L1/2)          input
      REAL*8         EW2(L2/2)          input
      REAL*8         AM1, AM2           input
      COMPLEX*16     CPL(2)             input
      COMPLEX*16     CE1(2,L1)          input
      COMPLEX*16     CE2(2,L2)          input
      REAL*8         PS1(4,L1/2+1)      input
      REAL*8         PS2(4,L2/2+1)      input
      REAL*8         EP(4,LV)           input
      INTEGER        LT(0:3)            output
      COMPLEX*16     AV(0:L1*L2*LV-1)   output
```

1. L1, L2 imply the freedom of fermion for U_1, U_2, respectively.
   It takes 2 for external line and 4 for internal line.
2. LV is, like LP in SMEXTV or SMINTV, the freedom of the polarization vector.
   external line : 2 for A_μ, G_μ and 3 for $W^\pm_\mu$, Z_μ.
   internal line : 3 for A_μ, G_μ and 4 for $W^\pm_\mu$, Z_μ.
3. The variables EW1, EW2 take the value EW created in SMINTF for internal line and (1) for particle or (-1) for anti-particle in the external line.
4. AM1, AM2 are masses of fermions.
5. CPL(1), CPL(2) represent left-handed(C_L), right-handed(C_R) coupling constant, respectively.
6. CE1, CE2 is CE created in SMINTF or SMEXTF.
7. PS1, PS2 is PS created in SMINTF or SMEXTF.

8. EP represents the polarization vector of vector boson and is the same one as created in SMEXTV or SMINTV.

9. AV the final result of amplitude(table).
   The ordering of index in the table is $U_1(p_1), \overline{U}_2(p_2)$, vector.

10. LT is the data which represent the structure of the table AV and used in SMCONF, SMCONV and AMPORD.

Fermion-fermion-scalar coupling

This calculates fermion-fermion-scalar vertex

$$\overline{U}_2(p_2)\left(C_L\frac{1-\gamma_5}{2}+C_R\frac{1+\gamma_5}{2}\right)U_1(p_1)$$

where U represents a spinor u or v. It stores the results for all the combinations of helicity and polarization (See section 6.1.11).

```
       CALL SMFFS(L1, L2, EW1, EW2, AM1, AM2, CPL, CE1, CE2,
     &            PS1, PS2, LT, AV)

       INTEGER        L1, L2          input
       REAL*8         EW1(L1/2)       input
       REAL*8         EW2(L2/2)       input
       REAL*8         AM1, AM2        input
       COMPLEX*16     CPL(2)          input
       COMPLEX*16     CE1(2,L1)       input
       COMPLEX*16     CE2(2,L2)       input
       REAL*8         PS1(4,L1/2+1)   input
       REAL*8         PS2(4,L2/2+1)   input
       INTEGER        LT(0:3)         output
       COMPLEX*16     AV(0:L1*L2-1)   output
```

1. L1, L2 are freedom of fermion corresponding to U_1, U_2, respectively.
   It takes 2 for external line and 4 for internal line.

2. EW1, EW2 is EW created in SMINTF for external line and (1) for particle or (−1) for anti-particle for internal line.

3. AM1, AM2 are masses of fermions.

4. CPL(1), CPL(2) represent left-handed (C_L), right-handed (C_R) coupling constants, respectively.

5. CE1, CE2 is CE created in SMINTF or SMEXTF.

6. PS1, PS2 is PS created in SMINTF or SMEXTF.
7. AV is the table of amplitude. The ordering of the index in the table is $U_1(p_1)$, $\overline{U}_2(p_2)$, scalar.
8. LT is the data which represents the structure of the table AV and used in SMCONF, SMCONV and AMPORD.

Vector-vector-vector coupling

This calculates 3-point vertex of vector bosons,

$$C[((p_1 - p_2).\epsilon_3)(\epsilon_1.\epsilon_2) + ((p_2 - p_3).\epsilon_1)(\epsilon_2.\epsilon_3) + ((p_3 - p_1).\epsilon_2)(\epsilon_3.\epsilon_1)]$$

(See section 6.1.3).

```
      CALL SMVVV(L1,L2,L3, K1,K2,K3, CPL,
     &           P1,P2,P3, E1,E2,E3, LT,AV)

      INTEGER        L1, L2, L3        input
      INTEGER        K1, K2, K3        input
      COMPLEX*16     CPL               input
      REAL*8         P1(4),P2(4),P3(4) input
      REAL*8         E1(4,L1)          input
      REAL*8         E2(4,L2)          input
      REAL*8         E3(4,L3)          input
      INTEGER        LT(0:3)           output
      COMPLEX*16     AV(0:L1*L2*L3-1)  output
```

1. L1, L2, L3 is, like LP in SMINTV or SMINTV, the freedom of polarization vector.
2. K1, K2, K3 denote the direction of the momentum.
   If momentum is incoming to vertex, then it takes 1, otherwise -1.
3. CPL is coupling constant C.
4. P1, P2, P3 are momenta of particles.
5. E1, E2, E3 are equal to EP created in SMEXTV or SMINTV and represent the vector of polarization vector.
6. AV is the resultant table of amplitude. The ordering of the index in the table is the same as that of the arguments, namely the order of vector-1, vector-2 and vector-3.
7. LT is the data which represents the structure of the table AV and is used in SMCONV and AMPORD.

Gluon-gluon-gluon coupling

This calculate gluon 3-point vertex

$$C[((p_1 - p_2).\epsilon_3)(\epsilon_1.\epsilon_2) + ((p_2 - p_3).\epsilon_1)(\epsilon_2.\epsilon_3) + ((p_3 - p_1).\epsilon_2)(\epsilon_3.\epsilon_1)].$$

```
      CALL SMGGG(L1,L2,L3, K1,K2,K3, CPL,
                 P1,P2,P3, E1,E2,E3, LT,AV)
```

This is the same as SMVVV.

1. L1, L2, L3 is, like LP created in SMINTV the freedom of polarization vector.
2. K1, K2, K3 denote the direction of momentum.
   If momentum is coming into the vertex, then it takes 1, otherwise -1.
3. CPL is the coupling constant C.
4. P1, P2, P3 are momenta of particles.
5. E1, E2, E3 is the polarization vector of vector boson and equal to EP created in SMINTV.
6. AV is the table of the resultant amplitude. The ordering of the index of the table is the same as the arguments, vector-1, vector-2 and vector-3.
7. LT is the data which represents the structure of the table AV and is used in SMCONV and AMPORD.

Scalar-vector-vector coupling

This calculates scalar-vector^2 vertex $C(\epsilon_2.\epsilon_3)$ (See section 6.1.7).

```
      CALL SMSVV(L2, L3, CPL, E2, E3, LT, AV)

      INTEGER        L2, L3          input
      COMPLEX*16     CPL             input
      REAL*8         E1(4,L2)        input
      REAL*8         E2(4,L3)        input
      INTEGER        LT(0:3)         output
      COMPLEX*16     AV(0:L2*L3-1)   output
```

1. L2, L3 are, like LP in SMINTV, the freedom of the polarization vector.

2. `CPL` is the coupling constant C.

3. `E2, E3` are polarization vectors of vector boson and are equal to `EP` created in `SMEXTV` or `SMINTV`.

4. `AV` is the table of resultant amplitude. The ordering of index in the table is the same as the arguments, scalar, vector-2 and vector-3.

5. `LT` is the data which represents the structure of the table `AV` and used in `SMCONV`, `SMCONS` and `AMPORD`.

Scalar-scalar-vector coupling

This calculates scalar^2-vector vertex $C((p_1 - p_2).\epsilon_3)$ (See section 6.1.6).

```
CALL SMSSV(L3, K1, K2, CPL, P1, P2, E3, LT, AV)

INTEGER        L3              input
INTEGER        K1, K2          input
COMPLEX*16     CPL             input
REAL*8         P1(4),P2(4)     input
REAL*8         E3(4,L3)        input
INTEGER        LT(0:3)         output
COMPLEX*16     AV(0:L3-1)      output
```

1. `L3` is, like `LP` in `SMINTV`, the freedom of polarization vector.

2. `K1, K2` denotes the direction of momentum of scalar.
   If momentum is coming into the vertex, it takes 1 and otherwise -1.

3. `CPL` is the coupling constant C.

4. `P1, P2` are momenta of scalar.

5. `E3` is polarization vector of vector boson and equal to `EP` created in `SMINTV`.

6. `AV` is the table of resultant amplitude. The order of index in the table is the same as the arguments, scalar-1, scalar-2 and vector.

7. `LT` is the data which represents the structure of the table `AV` and is used in `SMCONV`, `SMCONS` and `AMPORD`.

Scalar-scalar-scalar coupling

This calculates scalar^3 vertex C (See section 6.1.9).

```
CALL SMSSS(CPL, LT, AV)

COMPLEX*16     CPL             input
INTEGER        LT(0:3)         input
COMPLEX*16     AV(0:0)         output
```

1. CPL is the coupling constant C.
2. AV is the table of resultant amplitude. The order of index of the table is irrelevant.
3. LT is the data which represents the structure of the table AV and is used in SMCONS and AMPORD.

Vector-vector-vector-vector coupling

This calculates vector^4 vertex

$$c[(\epsilon_1.\epsilon_3)(\epsilon_2.\epsilon_4) + (\epsilon_1.\epsilon_4)(\epsilon_2.\epsilon_3) - 2(\epsilon_1.\epsilon_2)(\epsilon_3.\epsilon_4)]$$

(See section 6.1.4).

```
CALL SMVVVV(L1,L2,L3,L4,CPL,E1,E2,E3,E4,LT,AV)

INTEGER        L1, L2, L3, L4        input
COMPLEX*16     CPL                   input
REAL*8         E1(4,L1)              input
REAL*8         E2(4,L2)              input
REAL*8         E3(4,L3)              input
REAL*8         E4(4,L4)              input
INTEGER        LT(0:4)               output
COMPLEX*16     AV(0:L1*L2*L3*L4-1)   output
```

1. L1, L2, L3, L4 is, like LP in SMINTV, the freedom of polarization vector.
2. CPL is the coupling constant C.
3. E1, E2, E3, E4 is the polarization vector of vector boson and equal to EP created in SMINTV.
4. AV is the table of resultant amplitude. The ordering of index in the table is the same as the arguments 1, 2, 3 and 4.
5. LT is the data which represents the structure of the table AV and is used in SMCONV and AMPORD.

Gluon-gluon-gluon-gluon coupling

This is similar to the 4-point vertex of heavy vector boson, but it calculates only the coefficient of the single term of color factor

$$C[(\epsilon_1.\epsilon_3)(\epsilon_2.\epsilon_4) - (\epsilon_1.\epsilon_4)(\epsilon_2.\epsilon_3)].$$

```
CALL SMGGGG(L1,L2,L3,L4,CPL,E1,E2,E3,E4,LT,AV)
```

Arguments are the same as SMVVVV.

1. L1, L2, L3, L4 are, like LP created inSMINTV, the freedom of polarization vector.
2. CPL is the coupling constant C.
3. E1, E2, E3, E4 is the polarization vector of vector boson and equal to EP created in SMINTV.
4. AV is the table of the resultant amplitude. The ordering of the indices of the table is the same as the arguments, 1, 2, 3 and 4.
5. LT is the data which represents the structure of the table AV and is used in SMCONV and AMPORD.

Scalar-scalar-vector-vector coupling

This calculates scalar^2-vector^2 vertex $C(\epsilon_3.\epsilon_4)$ (See section 6.1.8).

```
CALL SMSSVV(L3,L4,CPL,E3,E4,LT,AV)

INTEGER        L3, L4          input
COMPLEX*16     CPL             input
REAL*8         E3(4,L3)        input
REAL*8         E4(4,L4)        input
INTEGER        LT(0:4)         output
COMPLEX*16     AV(0:L3*L4-1)   output
```

1. L3, L4 is, like LP created in SMINTV, the freedom of polarization vector.
2. CPL is the coupling constant C.
3. E3, E4 is the polarization vector of vector boson and equal to EP created in SMINTV.

4. AV is the table of resultant amplitude. The ordering of the index is the same as the arguments, scalar-1, scalar-2, vector-3 and vector-4.
5. LT is the data which represents the structure of the table AV and is used in SMCONV, SMCONS and AMPORD.

Scalar-scalar-scalar-scalar coupling

This calculates scalar^4 vertex C (See section 6.1.10).

```
CALL SMSSSS(CPL, LT, AV)

COMPLEX*16     CPL            input
INTEGER        LT(0:4)        input
COMPLEX*16     AV(0:0)        output
```

1. CPL is the coupling constant C.
2. AV is the table of resultant amplitude.
3. LT is the data which represent the structure of the table AV and is used in SMCONS and AMPORD. The ordering of the index in the table is irrelevant.

7.2.5 Connecting amplitude

The amplitude of a part of a given diagram, which is constructed by connecting smaller amplitudes by internal lines. The latter amplitudes are obtained by either calculating vertex or using the following subroutines. Called subroutines depend on the kinds of internal lines; for fermion line call SMCONF, for vector boson SMCONV and for scalar SMCONS.

```
CALL SMCONF(LT1,LT2,LP1,LP2,EW,AV1,AV2,LT,AV)
CALL SMCONV(LT1,LT2,LP1,LP2,EW,AV1,AV2,LT,AV)
CALL SMCONS(LT1,LT2,LP1,LP2,AV1,AV2,LT,AV)

INTEGER        LT1(0:*), LT2(0:*)           input
COMPLEX*16     AV1(0:*), AV2(0:*)           input
INTEGER        LP1, LP2                     input
REAL*8         EW(*)                        input
INTEGER        LT(0: number if indices)     input
COMPLEX*16     AV(0: freedom of index -1 )  output
```

1. AV1, AV2 are tables of resultant amplitude.

2. LT1, LT2 represent the structure of the table AV.
3. LP1, LP2 indicate which particles in LT1, LT2 are connected, respectively.
4. EW is identical with EW created in SMINTF, SMINTV.
5. AV is the table of resultant amplitude.
6. LT is the data which represents the structure of the table AV and is used in SMCONx or AMPORD. The ordering of the indices is the same as that of LT1, LT2 omitting the particles to be connected.

7.2.6 Check consistency of generated code

This subroutine checks consistency between generated code and library by comparing version numbers.

```
      CALL SMINIT(NV, NS)

      INTEGER     NV       input
      INTEGER     NS       input
```

1. NV, NS are version and sub-version number of grace system, by which source code is generated.

7.3 Program package CHANEL

7.3.1 Decomposition of propagator

Polarization vector

This sets components of polarization vectors.

```
CALL POLA(I, A, AM, P, EP, EM)

INTEGER        I              input
REAL*8         A              input
REAL*8         AM             input
REAL*8         P(4)           input
REAL*8         EP(4)          output
REAL*8         EM(4)          output
```

1. the expressions for rectangular polarization basis are presented in Eq.(2.163).
2. `I` : polarization state of vector boson.
3. `A` : gauge parameter of vector boson. `A` ≥ 100 gives the unitary gauge propagator for massive vector boson.
4. `AM` : mass of vector boson.
5. `P(4)`: momentum of vector boson.
6. `EP(4)` : polarization vector for state `I`.
7. `EM(4)` : weight factor, which are presented in Eq.(2.164).

Decomposition of four momentum

This decomposes the momentum of a massive fermion to two light-like vectors according to Eq.(2.140).

```
CALL SPLTQ(AM, P, P2, P1)

REAL*8         AM             input
REAL*8         P(4)           input
REAL*8         P1(4), P2(4)   output
```

1. `AM` : mass of fermion.
2. `P(4)` : momentum of fermion.
3. `P1(4)`,`P2(4)` : decomposed light-like vectors.

Phase factors of fermion

This calculates phase factors of the massive fermion presented in Eqs. (2.145) and (2.146).

```
CALL PHASEQ(I,P,C)

INTEGER        I              input
REAL*8         P(4)           input
COMPLEX*16     C(2)           output
```

1. `I` : `I=1` for $c_{\pm}(p)$ and `I=2` for complex conjugate of $c_{\pm}(p)$.
2. `P(4)` : momentum of massive fermion.
3. `C(2)` : calculated phase factors.

Split four momentum of internal fermion

This decomposes momentum of internal fermion with mass m into a light-like vector and time-like vector with momentum square m^2.

```
CALL SPLT(AM, P, S1, S2, P1, P2)

REAL*8         AM             input
REAL*8         P(4)           input
REAL*8         S1, S2         output
REAL*8         P1(4), P2(4)   output
```

1. `AM` : mass of internal fermion.
2. `P(4)` : momentum of internal fermion.
3. `S1,S2` : sign factors for the decomposed vectors.
4. `P1(4),P2(4)` : decomposed four vectors.

7.3.2 Vertices

Vector-massless fermion vertex

This calculates vertex amplitudes for the vector boson-massless fermions vertex presented in Eq.(2.135).

```
      CALL FFV0(P1,P2,P,AALL)

      REAL*8         P1(4), P2(4)    input
      REAL*8         P(4)            input
      COMPLEX*16     AALL(2)         output
```

1. the explicit expressions for specified k_0 are presented in Eq.(2.138).
2. `P1(4)`,`P2(4)` : momenta of massless fermions.
3. `P(4)` : polarization vector of vector boson coupled to fermion.
4. `AALL(2)` : calculated results of vertex amplitudes.

Vector-fermion vertex

This calculates vertex amplitude for the vector boson-massive fermions vertex presented in Eq.(2.148).

```
      CALL FFV(L,II,I,AAM,AM,AL,AR,CC,C,Q1,Q2,P1,P2,Q,AALL)

      INTEGER        L               input
      INTEGER        I, II           input
      REAL*8         AM, AAM         input
      REAL*8         AL, AR          input
      COMPLEX*16     C(2), CC(2)     input
      REAL*8         P1(4), P2(4)    input
      REAL*8         Q1(4), Q2(4)    input
      REAL*8         Q(4)            input
      COMPLEX*16     AALL(4,2,2)     output
```

1. `L` : polarization state of vector boson.
2. `I,II` : indices to specify fermion or antifermion state, where `I(II)`=3 for fermion and `I(II)`=1 for antifermion, respectively.
3. `AM,AAM` : masses of fermions.
4. `AL,AR` : coupling constants for vertex.
5. `C(2)`,`CC(2)` : phase factors for massive fermions.
6. `P1(4)`,`P2(4)`,`Q1(4)`,`Q2(4)` : light-like vectors decomposed by subroutine `SPLTQ`.
7. `Q(4)` : polarization vector of vector boson.
8. `AALL(4,2,2)` : calculated results of vertex amplitudes for all possible helicity states.

Scalar-massless fermion vertex

This calculates vertex amplitudes for scalar boson-massless fermions vertex presented in Eq.(2.151).

```
      CALL FFS0(P1, P2, AALL)

      REAL*8          P1(4), P2(4)     input
      COMPLEX*16      AALL(2)          output
```

1. `P1(4),P2(4)` : momenta of massless fermions.

2. `AALL(2)` : calculated results of vertex amplitudes.

Scalar-fermion vertex

This calculates vertex amplitudes for scalar boson-massive fermions vertex presented in Eq.(2.150).

```
      CALL FFS(II,I,AAM,AM,AL,AR,CC,C,Q1,Q2,P1,P2,AALL)

      INTEGER         I, II            input
      REAL*8          AM, AAM          input
      REAL*8          AL, AR           input
      COMPLEX*16      C, CC            input
      REAL*8          P1(4), P2(4)     input
      REAL*8          Q1(4), Q2(4)     input
      COMPLEX*16      AALL(2,2)        output
```

1. `I,II` : indices to specify fermion or antifermion state, where `I`(`II`)=3 for fermion and `I`(`II`)=1 for antifermion, respectively.

2. `AM,AAM` : masses of fermions.

3. `AL,AR` : coupling constants for vertex.

4. `C(2),CC(2)` : phase factors for massive fermions.

5. `P1(4)`, `P2(4)`, `Q1(4)`, `Q2(4)` : light-like vectors decomposed by subroutine `SPLTQ`.

6. `AALL(2,2)` : calculated results of vertex amplitudes for all possible helicity states.

Three vector vertex

This calculates vertex amplitude for three vector boson vertex, presented in Eq.(2.155).

```
CALL VVV(GG,P1,P2,P3,EP1,EP2,EP3,AALL)

REAL*8         GG               input
REAL*8         P1(4), P2(4)     input
REAL*8         P3(4)            input
REAL*8         EP1(4), EP2(4)   input
REAL*8         EP3(4)           input
REAL*8         AALL             output
```

1. the momenta of particles with vertices are taken to flow in.
2. `GG` : coupling constant for vertex.
3. `P1(4),P2(4),P3(4)` : momenta of the vector bosons.
4. `EP1(4),EP2(4),EP3(4)` : polarization vectors of vector bosons.
5. `AALL` : calculated result of vertex amplitude for given polarization states.

Four vector vertex

This calculates vertex amplitudes for four vector boson vertex, presented in Eq.(2.156).

```
CALL VVVV(GG,EP1,EP2,EP3,EP4,AALL)

REAL*8         GG               input
REAL*8         EP1(4), EP2(4)   input
REAL*8         EP3(4), EP4(4)   input
REAL*8         AALL             output
```

1. the momenta of particles with vertices are taken to flow in.
2. `GG` : coupling constant for vertex.
3. `EP1(4),EP2(4),EP3(4),EP4(4)` :polarization vectors of vector bosons.
4. `AALL` : calculated result of vertex amplitude for given polarization states.

Scalar-vector-vector vertex

This calculates vertex amplitudes for vector bosons-scalar boson vertex, presented in Eq.(2.158).

```
CALL VVS(GG,EP1,EP2,AALL)

REAL*8          GG               input
REAL*8          EP1(4), EP2(4)   input
REAL*8          AALL             output
```

1. the momenta of particles with vertex are taken to flow in. This subroutine can be used for $VVSS$ vertex.

2. `GG` : coupling constant for vertex.

3. `EP1(4),EP2(4)` : polarization vectors of vector bosons.

4. `AALL` : calculated result of vertex amplitude for given polarization states.

Scalar-scalar-vector vertex

This calculates vertex amplitudes for vector boson-scalar bosons vertex, presented in Eq.(2.157).

```
CALL SSV(GG,P1,P2,EP,AALL)

REAL*8          GG               input
REAL*8          P1(4), P2(4)     input
REAL*8          EP(4)            input
REAL*8          AALL             output
```

1. `GG` : coupling constant for vertex.

2. `P1(4),P2(4)` : momenta of vector bosons.

3. `EP(4)` : polarization vector of vector boson.

4. `AALL` : calculated result of vertex amplitude for given polarization states.

7.3.3 Effective vertices

Four fermion vertex

This calculate fermion-fermion-interactions mediated by a vector boson.

```
CALL FFFF(N, AJM1, AJM2, EM, AALL)

INTEGER         N               input
COMPLEX*16      AJM1(4,2,2)     input
COMPLEX*16      AJM2(4,2,2)     input
REAL*8          EM(4)           input
COMPLEX*16      AALL(2,2,2,2)   output
```

1. `N` : no. of polarization states for intermediated vector boson.
2. `AJM1(4,2,2), AJM2(4,2,2)` : vertex amplitudes for fermion-fermion-vector boson vertices calculated by subroutine `FFV`.
3. `EM(4)` : weight factors to reconstruct the numerator of the vector boson propagator calculated by subroutine `POLA`.
4. `AALL(2,2,2,2)` : calculated results for all possible helicity states for fermions.

Fermion-fermion-vector-vector vertex

This calculate fermion-fermion going to vector boson pair mediated by a vector boson.

```
CALL FFVV(N, AJMF, AJMV, EM, AALL)

INTEGER         N               input
COMPLEX*16      AJMF(4,2,2)     input
COMPLEX*16      AJMV(4,4,4)     input
REAL*8          EM(4)           input
COMPLEX*16      AALL(2,2,4,4)   output
```

1. `N` : no. of polarization states for intermediated vector boson.
2. `AJMF(4,2,2)` : vertex amplitude for fermion-fermion-vector boson vertex calculated by subroutine FFV.
3. `AJMV(4,4,4)` : vertex amplitude for three vector boson vertex calculated by subroutine VVV.

4. `EM(4)` : weight factors to reconstruct the numerator of the vector boson propagator calculated by subroutine `POLA`.

5. `AALL(2,2,4,4)` : calculated results for all possible helicity and polarization states.

When the user combines his own subprograms with generated source programs of integration and event generation, the user should be careful not to use the names of subprograms and named common block, because the program does not work when the user uses the same names as predefined ones in this system.

The list of the names of subprograms generated or internally used, is shown in List A.1, and the list of the names of common block generated or internally used, is shown in List A.2.

AMnnnn	BSDATE	BSUNIX	FFV	SHUPDT	SMINIV	SPLTQ	XHGRID
AMPARM	BSDSUM	BSUSRI	FFV0	SMCONF	SMPRPD	SPMAIN	XHINIT
AMPORD	BSGDEF	BSUTIM	FUNC	SMCONS	SMSSS	SPRING	XHORDR
AMPSUM	BSINIT	BSWRIT	KINEM	SMCONV	SMSSSS	SPTERM	XHPLOT
AMPTBL	BSISUM	DHFILL	KINIT	SMEXTF	SMSSV	SSV	XHRNGE
BASES	BSLIST	DHINIT	PHASEQ	SMEXTV	SMSSVV	USERIN	XHSAVE
BHINIT	BSMAIN	DHPLOT	POL	SMFFS	SMSVV	USROUT	XHSCLE
BHPLOT	BSORDR	DRLOOP	POLA	SMFFV	SMVVV	VVS	
BHRSET	BSPRNT	DRN	SETMAS	SMGGG	SMVVVV	VVV	
BHSAVE	BSREAD	DRNSET	SHCLER	SMGGGG	SPEVNT	VVVV	
BHSUM	BSTCNV	FFS	SHPLOT	SMINIF	SPINIT	XHCHCK	
BSCAST	BSTIME	FFS0	SHRSET	SMINIT	SPLT	XHFILL	

List A.1 A list of the names of subprograms

AMnnnn means the corresponding subroutines to the nnnn-th graph. When 28 graphs are generated, then nnnn varies from 0001 to 0028.

AMCNST	AMMASS	AMWORK	BASE4	BSRSLT	NINFO	SMDBGG	SPRNG1
AMCPLC	AMREG	BASE0	BASE5	BSWORK	PLOTB	SMEXTP	XHCNTL
AMEXTR	AMSLCT	BASE1	BASE6	BTIME	PLOTH	SMGAUG	
AMGMMA	AMSPIN	BASE2	BDATE	LOOP0	RANDM	SMGAUS	
AMGRPH	AMWORI	BASE3	BSCNTL	LOOP1	SMATBL	SP4VEC	

List A.2 A list of names of named common block

In the following index, the mark “(v)” is for vector version.

`AMnnnn`	calculates amplitude of the *nnnn*-th graph, 78, **87**, 165(v), **169**(v)
`AMPARM`	defines of coupling constants and others, 78, **82**, 165(v)
`AMPORD`	arranges amplitudes, 78, **88**
`AMPSUM`	sums matrix elements over the helicity states, 78, 165(v)
`AMPTBL`	calls `AMnnnn` to calculate amplitudes, 78, **85**, 165(v), **167**(v)
`BHINIT`	initializes the numbers of histograms and scatter plots, 111, **120**
`DHFILL`	fills scatter plots, 115, **121**
`DHINIT`	initializes scatter plots, 111, **121**
`DVFILL`	fills scatter plots, **178**(v)
`FFFF`	calculates four fermion vertex, **231**
`FFS`	calculates scalar-fermion vertex, **228**
`FFS0`	calculates scalar-massless fermion vertex, **228**
`FFV`	calculates vector-fermion vertex, **227**
`FFV0`	calculates vector-massless fermion vertex, **226**
`FFVV`	calculates fermion-fermion-vector-vector vertex, **231**
`FUNC`	calculates the numerical values of differential cross section, 78, **114**, 134
`INCL1`	defines the common variables for masses, amplitude tables *etc*, 78, **83**, 165(v), **166**(v)
`INCL2`	defines the work space for `AMPTBL`, 78, 165(v)
`INCLH`	defines the number of histograms and scatter plots, 78, **109**, 165(v), 166(v)
`INCLVB`	defines the common for integration variables *etc*, 165(v), 166(v), **172**(v), 176(v)
`INCLVS`	defines the vector length, 165(v), 166(v), **172**(v)
`KINEM`	derives particle four momenta from the integration variables, 78, **93**, 165(v), **179**(v)
`KINIT`	initializes kinematics, 78, **90**, 165(v), 166(v), 171(v)

- **Getting system and information**
  Any request on the system should be mailed to `grace@minami.kek.jp`.

- **Changing Lagrangian**
  `GRACE` system has almost no freedom to change Lagrangian except for deleting particles or interactions. For introduction of a new type of interaction, one must prepare a FORTRAN subroutine, which calculates helicity amplitudes of the vertex in a similar way to `CHANEL` library. Moreover, modifications of the FORTRAN source code generator is necessary in order to include corresponding subroutine calls in the generated code.

- **Testing generated code**
  There are several possibilities that the generated FORTRAN code produces false result (see section 2.6 p.55 and chapter 3 p.72, 96, 101). Especially one must pay attention to numerical cancellation. It is desirable to compare with the result of calculation in quadruple precision.

- **Gauge parameter**
  The way of specifying the gauge parameter in the generated FORTRAN code is slightly different from one in `CHANEL` library. Unitary gauge is selected in `CHANEL` library when gauge parameter is set to a greater value than 100. However, in the generated code, unitary gauge is selected by another variable (see section 3.2.1), with which denominators of propagators are correctly calculated and irrelevant graphs are omitted from the calculation.

- **Decay width**
  Since gauge invariance and gauge cancellation is violated by non-zero value of decay width of particles, unexpectedly large value of cross section may be obtained at higher energy (see section 2.6).

- **Strong coupling constant**
  Strong coupling constant $\alpha_s(Q^2)$ should be supplied in `kinem` even if it is taken to be constant(see section 2.3.3).

- **Quark mixing matrix**
  In the present version of `GRACE`, the mixing matrix is unity(see section 2.3.2).

- **Numerical instability**
  When the considered process contains a virtual photon exchanged in the t-channel, such as peripheral diagrams in $e^+e^- \to e^+e^-\mu^+\mu^-$ or $e^+e^- \to e^+e^-\gamma$ and the phase space allows extremely small values to the virtual photon mass, serious instability will take place in the numerical calculation(see section 2.6).

- **Convergence behavior of numerical integration**
  The accuracy of each iteration must be stable in the integration step. When the integration variables does not suit for the integrand, it is unstable or fluctuate iteration by iteration and jumps suddenly to a big value in the worst case (see sections 2.6 and 3.5.6).

- **Event generation**
  If the number trial distribution has a long tail, the efficiency of event generation is poor(see 3.6.4).

Bibliography

[1] K. Aoki *et al.*, *Suppl. Prog. Theor. Phys.* **73**, 1982.

[2] T. Muta, *"Foundations of Quantum Chromodynamics"*, World Scientific, 1987.

[3] T. Kaneko, in *"New Computing Techniques in Physics Research"*, ed. D. Perret-Gallix and W. Wojcik, p.555, 1990, Edition du CNRS, Paris,
T. Kaneko and H. Tanaka, in *"Proc. of the Second Workshop on JLC"*, ed. S. Kawabata, p.250, 1991, KEK Proceeding 91-10,
T. Kaneko *et al.*, in *"New Computing Techniques in Physics Research II"*, ed. D. Perret-Gallix, p.659, 1992, Edition du CNRS, Paris.

[4] H. Tanaka, *Comput. Phys. Commun.* **58**(1990)153.

[5] H. Tanaka, T. Kaneko and Y. Shimizu, *Comput. Phys. Commun.* **64**(1991)149.

[6] S. Kawabata, *Comput. Phys. Commun.* **41**(1986)127.

[7] See for example,
F. Harary, *"Graph Theory"*, Addison-Wesley, Reading, Massachusetts, 1972.
For enumeration of various classes of graphs,
F. Harary and E.M. Palmer, *"Graphical Enumeration"*, Academic Press, New York, 1973.
For computer algorithms,
A.V. Aho, J.E. Hopcroft and J.D. Ullman, *"The Design and analysis of Computer Algorithms"*, Addison-Wesley, Reading, Massachusetts, 1972;
L. Kučera, *"Combinatorial Algorithms"*, Adam Hilger, Bristol, 1990.

[8] J.Fujimoto *et al.*, *Suppl. Prog. Theor. Phys.* **100**, 1991.

[9] T. Kaneko and S. Kawabata, *Comput. Phys. Commun.* **55** (1989) 141.