GRC4F: User-Guide

Version 2.2(41)
25 Jan. 2001

Preliminary

GRC4F version 2.2

Jan, 25 2001

Summary

GRC4F is a four fermion event generator with the following features:

- Non zero fermion mass
- Selection of the initial and final state helicity
- Several ISR options: SF or QEDPSi and QEDPDif (n-photon are generated with the proper angular distribution).
- Proper treatment of the gauge violation induced by the W width. This implies that events can be generated with no cuts
- All 76 processes with all diagrams are included.

Version 2 is essentially a new interface to the “grc4f” kernel with the following additions:

- All processes (76) can be generated by a single executable
- But a given process can be generated by “making” a dedicated single process smaller executable.
- Only 2 subroutines should be known to the user:
Subroutine grcevt(...) Event steering routine
Subroutine grcpar(...) Setting parameters
- Any parameter modification can be automatically printed
- Generic (or inclusive) process has been implemented, for example: process: ’CLNZUPDP’ will generate all semi-leptonic channels (6 at LEP) and each for both charge combinations.
- For each process, sets of diagrams can be selected (see parameter ’set’ for example set=1 select only WW resonant diagrams)
- Final state event helicity is made available for the 4 fermions, further helicity dependent decay (ex. tau) will be introduced in the next release. However ISR photon helicity is left undetermined.
- Process dependent integration parameters are now included by default (ncall, itmx1, . . .).
- New improved kinematics covering the full phase space (in particular for the 2 photon physics)

Limitations:

- No CKM mixing, namely no W decays in “cd” or “us”

Difference between version 2.2 and version 2.1

- QEDPS-tchanel is supported. for $e^- e^+ \rightarrow e^- \ \bar{\nu_e} \ \bar{\mu} \ \nu_\mu$, $e^- \ \bar{\nu_e} \ \bar{\tau} \ \nu_\tau$, $e^- \ e^+ \ \mu \ \bar{\mu}$, $e^- \ e^+ \ \tau \ \bar{\tau}$, $e^- \ \bar{\nu_e} \ u \ \bar{d}$, $e^- \ \bar{\nu_e} \ c \ \bar{s}$, $e^- \ e^+ \ q \ \bar{q}, (q = u, d, c, s, b)$.
- the random number for QEDPS is independently generated from BASES/SPRING. Then there is no compatibility of **bases.data** between 2.2 and before.

Difference between version 2.0 and version 2.1

The internal parameter in BASES/SPRING (Integration/Event Generation) was changed. (See subsection 8.2). One will get higher efficiency in Event Generation since larger information in the integraion step, is used.

The user interface is not changed. Some trivial bugs has been removed.

Reference to grc4f

The reference to grc4f is

J.Fujimoto, et.al. Comput. Phys. Commun. 100 (1997) 128.

The references to QEDPS are as follows:

- Y. Kurihara, et.al. Prog. Theor. Phys. 103 (200) 1199.
- Y. Kurihara, et.al. **hep-ph/0011276**.

1 Introduction

The package provides essentially two interface subroutines to build any general purpose 4 fermion generator:

- Subroutine grcevt(...) Event steering routine
- Subroutine grcpar(...) Setting parameters.

It is the user responsibility to build the main program depending on its needs and on the specific experimental requirements. However some implementation examples are included in this package in the directory “examples”.

2 Naming Convention particles and processes

This convention allows to mix particles and generics in the process name definition. A ’generic’ represents a set of particles, like for example: all leptons or all charged leptons.
A process is defined as a set of 4 particles (generic or single).
Each particle name is made of 2 letters where the case (upper or lower) is meaningful.

2.1 Elementary Particle Definition

2.1.1 Leptons

ec	e^-	mc	μ^-	tc	τ^-	en	ν_e	mn	ν_μ	tn	ν_τ
eC	e^+	mC	μ^+	tC	τ^+	eN	$\bar{\nu_e}$	mN	$\bar{\nu_\mu}$	tN	$\bar{\nu_\tau}$

2.1.2 Quarks

uq	u	cq	c	tq	t	dq	d	sq	s	bq	b
uQ	$\bar{u}$	cQ	$\bar{c}$	tQ	$\bar{t}$	dQ	$\bar{d}$	sQ	$\bar{s}$	bQ	$\bar{b}$

2.1.3 Generic names

Cc	e^-, μ^-, τ^-	CC	e^+, μ^+, τ^+	CL	e^-, μ^-, τ^-, e^+, μ^+, τ^+
Bc	μ^-, τ^-	BC	μ^+, τ^+		
eL	e^-, e^+	mL	μ^-, μ^+	tL	τ^-, τ^+
Nn	ν_e, ν_μ, ν_τ	NN	$\bar{\nu_e}$, $\bar{\nu_\mu}$, $\bar{\nu_\tau}$	NZ	ν_e, ν_μ, ν_τ, $\bar{\nu_e}$, $\bar{\nu_\mu}$, $\bar{\nu_\tau}$

LP	e^-, μ^-, τ^-, e^+, μ^+, τ^+, ν_e, ν_μ, ν_τ, $\bar{\nu_e}$, $\bar{\nu_\mu}$, $\bar{\nu_\tau}$
KP	μ^-, τ^-, μ^+, τ^+, ν_e, ν_μ, ν_τ, $\bar{\nu_e}$, $\bar{\nu_\mu}$, $\bar{\nu_\tau}$

`uP`	u,$\bar{u}$	`cP`	c,$\bar{c}$	`tP`	t,$\bar{t}$
`dP`	d,$\bar{d}$	`sP`	s,$\bar{s}$	`bP`	b,$\bar{b}$
`Uq`	u,c,t	`UQ`	$\bar{u}$,$\bar{c}$,$\bar{t}$		
`Dq`	d,s,b	`DQ`	$\bar{d}$,$\bar{s}$,$\bar{b}$		
`UP`	u,c,t,$\bar{u}$,$\bar{c}$,$\bar{t}$	`DP`	d,s,b,$\bar{d}$,$\bar{s}$,$\bar{b}$		
`Qq`	u,c,t,d,s,b	`QQ`	$\bar{u}$,$\bar{c}$,$\bar{t}$, $\bar{d}$,$\bar{s}$,$\bar{b}$		
`QP`	u,c,t,d,s,b, $\bar{u}$,$\bar{c}$,$\bar{t}$, $\bar{d}$,$\bar{s}$,$\bar{b}$				

`FP`	e^-, μ^-, τ^-, e^+, μ^+, τ^+, ν_e, ν_μ, ν_τ, $\bar{\nu_e}$, $\bar{\nu_\mu}$, $\bar{\nu_\tau}$, u,c,t,d,s,b, $\bar{u}$,$\bar{c}$,$\bar{t}$, $\bar{d}$,$\bar{s}$,$\bar{b}$
`EP`	μ^-, τ^-, μ^+, τ^+, ν_e, ν_μ, ν_τ, $\bar{\nu_e}$, $\bar{\nu_\mu}$, $\bar{\nu_\tau}$, u,c,t,d,s,b, $\bar{u}$,$\bar{c}$,$\bar{t}$, $\bar{d}$,$\bar{s}$,$\bar{b}$

2.2 Process name

The name of process is composed by 4 final particle names, i.e., 8 characters. For example,

$e^-e^+ \to e^-\ \bar{\nu}_e\ u\ \bar{d}$ `eceNuqdQ`

2.2.1 particle name ordering

In a process name, the particle ordering as given by the user is irrelevant for the process selection.

`eceNuqdQ`, `eNecuqdQ`, `eNdQecuq`, `uqdQeceN`, *etc.* $\Rightarrow$ `eceNuqdQ`

However the cuts refer to a fix particle numbering (particle number 3456, 1 and 2 being the initial particles) in the order given in the process list shown hereafter. for example:

`eceCeneN` $\to$ `ec` (3) `eC` (4) `en` (5) `eN` (6)

The user must be particularly cautious for the generic case, where one set of cuts applies to several processes. He/she should make sure that the ordering imposed by the system is consistent with the cuts he/she wants to apply.

In the case of final state with identical particles, the ordering is from the larger to the smaller particle energy (energy or momentum ordering)

2.2.2 conjugate process

$e^-e^+ \to e^+\ \nu_e\ \bar{u}\ d$

is a charge conjugate process of

$e^-e^+ \to e^-\ \bar{\nu}_e\ u\ \bar{d}$.

The charge cojugate process `eCenuQdq` is treated as the original process `eceNuqdQ` in the integration step. The cross section value is calucalted by `eceNuqdQ`. The total cross section of this process are multiplied by "a charge conjugate factor"(= 2).

For event generation step, the charge cojugate final states are generated too.

2.2.3 generic name

The gereric name specifies multi-processes, for example:

`eceNUqDQ` = `eceNuqdQ` + `eceNcqsQ`

`CLNZUPDP` = `eceNuqdQ` + `eceNcqsQ` + `mcmNuqdQ` + `mcmNcqsQ` + `tctNuqdQ` + `tctNcqsQ`

`FPFPFPFP` = all 4 fermion (76 processes)

2.3 Process Internal name

Process #	Process name	C	Process #	Process name	C
(1)	`eceCeneN`	1	(2)	`eceNmCmn`	2
(3)	`eceNtCtn`	2	(4)	`mnmCmcmN`	1
(5)	`tntCtctN`	1	(6)	`mcmNtCtn`	2
(7)	`eceCeceC`	1	(8)	`eceCmcmC`	1
(9)	`eceCtctC`	1	(10)	`mcmCmcmC`	1
(11)	`tctCtctC`	1	(12)	`mcmCtctC`	1
(13)	`eceCmnmN`	1	(14)	`eceCtntN`	1
(15)	`eneNmcmC`	1	(16)	`eneNtctC`	1
(17)	`tntNmcmC`	1	(18)	`mnmNtctC`	1
(19)	`eneNeneN`	1	(20)	`eneNmnmN`	1
(21)	`eneNtntN`	1	(22)	`mnmNmnmN`	1
(23)	`tntNtntN`	1	(24)	`tntNmnmN`	1

Process #	Process name	C	Process #	Process name	C
(25)	`eceNuqdQ`	2	(26)	`eceNcqsQ`	2
(27)	`mcmNuqdQ`	2	(28)	`mcmNcqsQ`	2
(29)	`tctNuqdQ`	2	(30)	`tctNcqsQ`	2
(31)	`eceCuquQ`	1	(32)	`eceCcqcQ`	1
(33)	`eceCdqdQ`	1	(34)	`eceCsqsQ`	1
(35)	`eceCbqbQ`	1	(36)	`mcmCuquQ`	1
(37)	`mcmCcqcQ`	1	(38)	`tctCuquQ`	1
(39)	`tctCcqcQ`	1	(40)	`mcmCdqdQ`	1
(41)	`mcmCsqsQ`	1	(42)	`mcmCbqbQ`	1
(43)	`tctCdqdQ`	1	(44)	`tctCsqsQ`	1
(45)	`tctCbqbQ`	1	(46)	`eneNuquQ`	1
(47)	`eneNcqcQ`	1	(48)	`eneNdqdQ`	1
(49)	`eneNsqsQ`	1	(50)	`eneNbqbQ`	1
(51)	`mnmNuquQ`	1	(52)	`mnmNcqcQ`	1
(53)	`tntNuquQ`	1	(54)	`tntNcqcQ`	1
(55)	`mnmNdqdQ`	1	(56)	`mnmNsqsQ`	1
(57)	`mnmNbqbQ`	1	(58)	`tntNdqdQ`	1
(59)	`tntNsqsQ`	1	(60)	`tntNbqbQ`	1

Process #	Process name	C	Process #	Process name	C
(61)	`uqdQdquQ`	1	(62)	`cqsQsqcQ`	1
(63)	`uqdQsqcQ`	2	(64)	`uquQuquQ`	1
(65)	`cqcQcqcQ`	1	(66)	`dqdQdqdQ`	1
(67)	`sqsQsqsQ`	1	(68)	`bqbQbqbQ`	1
(69)	`uquQcqcQ`	1	(70)	`uquQsqsQ`	1
(71)	`uquQbqbQ`	1	(72)	`cqcQdqdQ`	1
(73)	`cqcQbqbQ`	1	(74)	`dqdQsqsQ`	1
(75)	`dqdQbqbQ`	1	(76)	`sqsQbqbQ`	1

"C" means a charge conjugate factor, i.e.

C = 1 self conjugate process
C = 2 not self conjugate process

3 Information

3.1 Process and Cross section

Process information is made available in the common:

```
parameter( MPROC = 76 )
double precision xscton,xscter
common /grcusr/ xscton(MPROC), xscter(MPROC)
common /grcusi/ mulprc(0:MPROC), ichgcj(MPROC), mevent(MPROC)
```

The number of process is stored in `mulprc(0)`. The process numbers are stored in `mulprc(`k`)` $k = 1 \sim$ `mulprc(0)`. The charge conjugate factors for each process are stored in `ichgcj(`k`)` $k = 1 \sim$ `mulprc(0)`.

After integration step, the cross section values and stardard deviations are stored in `xscton(`k`)`, `xscter(`k`)` $k = 1 \sim$ `mulprc(0)`.

After event generation step, the number of events for each process are stored in `mevent(`k`)` $k = 1 \sim$ `mulprc(0)`.

3.2 Particle information

Particle information for event generation are stored in the common `LUJETS` of JETSET.

3.3 Particle Helicity information

Particle Helicity information for each fermion is made available in the common:

```
INTEGER LECT
COMMON/HELICITY/LECT(4000)
     LECT(I)= Helicity*2
            =  +1,    -1    right and left handed (fermion)
            =  +2, 0, -2    (bosons)
            =  999          undetermined
            = -999          not defined
```

Phonetically Helicity spells LECT. The subroutine `grchel` prints the particle helicity of the current event.

3.4 How to get Process name

The process name (`character*8`) is obtained by call the function `grcpcn(ith)`, where the argument `ith` means i-th process in array `mulprc`.

4 Simple examples

4.1 Single process

The following example for single process is probably the shortest program one can build to run “grc4f”:

```
c-----------------------------------------------------------------------
c     test for single process job (examples/ex1/grc4f.f)
c-----------------------------------------------------------------------
      program single
      implicit real*8 (a-h,o-z)
c-----------------------------------------------------------------------
      parameter( MPROC = 76 )
      common /grcusr/ xscton(MPROC), xscter(MPROC)
      common /grcusi/ mulprc(0:MPROC), ichgcj(MPROC), mevent(MPROC)
      character*8 grcpcn
      external    grcpcn
c-----------------------------------------------------------------------
c    1. Global initialization
c-----------------------------------------------------------------------
      call grcevt(-3,iopt,ier)
c-----------------------------------------------------------------------
c    2. Set parameters
c-----------------------------------------------------------------------
      call grcpar('ncall',40000,1,ier)
      call grcpar('itmx1',7,1,ier)
      call grcpar('itmx2',15,1,ier)
      call grcpar('isrtype',0,1,ier)
*     call grcpar('shower',0,1,ier)
c-----------------------------------------------------------------------
c    3. Process initilization
c-----------------------------------------------------------------------
      call grcevt(-2,iopt,ier)
c-----------------------------------------------------------------------
c    4. Integration step
c-----------------------------------------------------------------------
      call grcevt(-1,iopt,ier)
c-----------------------------------------------------------------------
c    5. Generation of 1 event
c-----------------------------------------------------------------------
      call grcevt(0,iopt,ier)
      if(ier.ne.0) then
         print *,'Error in grcevt(0): ',ier
      else
        call lulist(1)
        call grchel
      endif
c-----------------------------------------------------------------------
c    6. event summary
c-----------------------------------------------------------------------
      call grcevt(+1,iopt,ier)
c-----------------------------------------------------------------------
c    7. user  summary
c-----------------------------------------------------------------------
```

```
      do 1000 m = 1 , mulprc(0)
      write(6,6000) grcpcn(m),ichgcj(m),xscton(m), xscter(m)
 1000 continue
      stop
 6000 format(' Proc:',a,' ',i1,' x',e15.7,' +- ',e15.7)
      end
```

Description:

1. Global initialization
   All parameters are set to their default values.

2. Set parameters
   Any user specific parameter value should be set after this call and before the process initialization by call to grcpar.

3. Process initilization
   All process dependent parameter are internally set. After this call the user may get the process name or the default value of the process cross-section.

4. Integration step
   After this call the value of the cross-section is known for the given set of cuts.

5. Generation of 1 event
   Each call to `grcevt(0,...)` will generate a new event. `lulist` are the jetset routine to print one event and `grchel` prints the event helicity data.

6. event summary
   Event ummary print-out.

7. user summary
   The cross section value is printed.

4.2 Multi processes

```
*-----------------------------------------------------------------------
*     multi process (examples/ex2/grc4f.f)
*
*        %make
*        %grc4f < grc4f.d
*-----------------------------------------------------------------------
      program multi
      implicit real*8 (a-h,o-z)
      parameter( MPROC = 76 )
      common /grcusr/ xscton(MPROC), xscter(MPROC)
      common /grcusi/ mulprc(0:MPROC), ichgcj(MPROC), mevent(MPROC)
      character*8 grcpcn
      external    grcpcn
*-----------------------------------------------------------------------
*     Local variables
*-----------------------------------------------------------------------
```

```
      character*256 fbsda,fevt,fdum
      integer nevt(MPROC)
      data Nevent/100000/
*-----------------------------------------------------------------------
*   1. Global initialization to default value
*-----------------------------------------------------------------------
      call grcevt(-3,iopt,ierr)
*-----------------------------------------------------------------------
*   2. Set parameters
*-----------------------------------------------------------------------
* =============================================
*     process name
* =============================================
      call grcpar('process','CLNZUPDP',1,ierr)
*
*     call grcpar('amw',80.0d0,1,ierr)
*     if(ierr.ne.0) goto 1100
*     call grcpar('set',1,1,ierr)
*     if(ierr.ne.0) goto 1100
      call grcpar('canon',0,1,ierr)
      if(ierr.ne.0) goto 1100
* =============================================
* read additional parameter setting
* set ISR type no ISR=0 sf=1 qedps=2 qedpsi=3
* =============================================
      read *,id1
      call grcpar('isrtype',id1,1,ierr)
      call grcpar('ncall',40000,1,ierr)
      call grcpar('itmx1',7,1,ierr)
      call grcpar('itmx2',15,1,ierr)
      if(ierr.eq.1) goto 1100
*-----------------------------------------------------------------------
*   3.  Set logical unit for 'BASES DATA' file
*       (used for Event Generation for each process)
*-----------------------------------------------------------------------
      lubsdt = 31
      call grcpar('lubsdt',lubsdt,1,ierr)
      if(ierr.ne.0) goto 1100
*-----------------------------------------------------------------------
*   4.  Set logical unit for Result files and Open them.
*-----------------------------------------------------------------------
      lubsrl = 39
      call grcpar('lubsrl',lubsrl,1,ierr)
      if(ierr.ne.0) goto 1100
      open(lubsrl,file='bases.result',status='unknown',
     &            form='formatted',iostat=iostat)
      if( iostat .ne. 0 ) then
          print *,' Open file error for bases.result'
```

```
          stop
      endif
      lusprl = 40
      call grcpar('lusprl',lusprl,1,ierr)
      if(ierr.ne.0) goto 1100
      open(lusprl,file='spring.result',status='unknown',
     &            form='formatted',iostat=iostat)
      if( iostat .ne. 0 ) then
          print *,' Open file error for spring.result'
          stop
      endif
*------------------------------------------------------------------------
*   5. Process initialization
*      hereafter we cannot change any parameters.
*------------------------------------------------------------------------
      call grcevt(-2,iopt,ierr)
*------------------------------------------------------------------------
*   6.            Integration step
*          ( bases information files are generated for each process.)
*------------------------------------------------------------------------

      do 3100 m = 1 , mulprc(0)
          fbsda = grcpcn(m)//'.dat'
          fdum  = fbsda
          lu    = lubsdt
          open(lubsdt,file=fbsda,status='unknown',
     &            form='unformatted',err=1200)
          call grcevt(-1,m,ierr)
          close(lubsdt)
 3100 continue

*------------------------------------------------------------------------
* ==============================
* set specific jetset parameters
* ==============================
* Photon radiation off quarks
      CALL LUGIVE('MSTJ( 41) = 2')
* No jetset Initial state radiation
      CALL LUGIVE('MSTJ(107) = 0')

*------------------------------------------------------------------------
*   7.  Open Event file ("wlist.f" are used.)
*------------------------------------------------------------------------
      fevt = 'event.out'
      read(5,'(a)') fevt
      if( fevt.ne.' ')then
          lu=38
          fdum=fevt
```

```
          open(lu,file=fevt,status='unknown',
     &            form='unformatted',err=1200)
      endif
*-----------------------------------------------------------------------
*   8.  Calculation of Total cross section
*-----------------------------------------------------------------------
      totalx = 0.0d0
      do 4100 m = 1 , mulprc(0)
         totalx = totalx + ichgcj(m)*xscton(m)
 4100 continue

*-----------------------------------------------------------------------
*   9.  Event generation step for each process.
*       nevt(m): the number of events for each process.
*-----------------------------------------------------------------------
      do 5200 m=1,mulprc(0)
         nevt(m) = nint(ichgcj(m)*xscton(m)/totalx*Nevent)
         fbsda = grcpcn(m)//'.dat'
         fdum  = fbsda
         lu    = lubsdt
*================================
* open 'bases information file'
*================================
         open(lubsdt,file=fbsda,status='old',
     &            form='unformatted',err=1200)
*
         do 5100 i=1,nevt(m)
            call grcevt(0,m,ierr)
            if(ierr.ne.0) goto 5100
            if(i.le.10) then
               call lulist(1)
*              call grchel
            endif
            if(lu.ne.0)then
*====================
* write event on file
*====================
               call wlist(lu,ic)
               if(ic.eq.1) goto 5300
            endif
 5100    continue
         close(lubsdt)

 5200 continue
*==================
* end of generation
*==================
```

```
 5300 continue
*-----------------------------------------------------------------------
*  10.  Event summary (Don't forget)
*-----------------------------------------------------------------------
      call grcevt(1,iopt,ierr)
*-----------------------------------------------------------------------
*  11.  closing files
*-----------------------------------------------------------------------
      close(lu)
      close(lusprl)
      close(lubsrl)
*-----------------------------------------------------------------------
*  12.  user summary
*-----------------------------------------------------------------------
      do 2000 m = 1 , mulprc(0)
      write(6,6000) grcpcn(m),ichgcj(m),xscton(m), xscter(m), mevent(m)
 2000 continue
 6000 format(' Process: ',a8,' ',i2,' x  X-sect.(pb) ',d12.5,
     &       ' +- ',d12.5,' ',i5)
      stop
*-----------------------------------------------------------------------
 1100 print *,'Error in parameter setting'
      stop
*-----------------------------------------------------------------------
 1200 continue
      write(6,*)' Error in opening l.U.',lu
      write(6,*)' File: ',fdum
      stop
      end
```

Description:

1. Global initialization
   All parameters are set to their default values.
2. Set parameters
3. Set logical unit for BASES Information file.
   BASES Information files for each process are generated in the integration step. These files are used in the event generation step. In the case of multi-process, these files should be prepared by users.
4. Set logical unit for Result files and Open them.
5. Process initilization
   All process dependent parameter are internally set. After this call the user may get the process name or the default value of the process cross-section.
6. Integration step
   After this call the value of the cross-section is known for the given set of cuts.

```
do 3100 m = 1 , mulprc(0)
```

is do-loop to change the process. For each process, the different files of "BASES Information file" should be produced.

The each integration is executed by

```
call grcevt(-1,m,ierr)
```

. the second argument of `grcevt` should be the loop-index, i.e., "`m`".

7. Open Event file
   As an example, a event file will be opened. In `wlist.f`, one record are written. This event file will be analyzed by an utiltity program `checkevent` as one example.

8. Calculation of Total cross section
   Total cross section is calculated by using array `ichgcj` and `xscton`.

9. Event generation step

   ```
   do 5200 m=1,mulprc(0)
   ```

   is do-loop to change the process. For each process, the different files of "BASES Information file" should be opened.

   The numbers of event generated for each process are calculated using the cross section of each process, total cross section and a given event number:

   ```
   nevt(m) = nint(ichgcj(m)*xscton(m)/totalx*Nevent)

   do 5100 i=1,nevt(m)
   ```

   is do-loop for event generation for each process.

   One event generation is executed by

   ```
   call grcevt(0,m,ierr)
   ```

   . the second argument of `grcevt` should be the loop-index, i.e., "`m`".

   `lulist` are the jetset routine to print one event and `grchel` prints the event helicity data.

   In `wlist.f`, one record is written in event file.

10. event summary
    Event ummary print-out.

11. closin files

12. user summary
    The cross section value is printed.

5 Subroutine description

5.1 subroutine grcevt(iflag,iopt,ierr)

PURPOSE:
Main steering subroutine for BASES (integration) and SPRING (event generation)
PARAMETERS:

iflag:	`integer:`	Control flag
iopt:	`integer:`	Option
ierr :	`integer:`	Return code

iflag : integer

- -3 “Grc4f” external parameters default parameter initialization
- -2 Process parameter initialization
- -1 Integration
- 0 Event Generation
- +1 End of generation

In return, the parameter *ierr* contains error status: *ierr : integer*

- 0 OK
- 1 Error reading the temporary file
- 2 Error writing the temporary file
- 3 All event from the temporary file have been used.

iopt: integer

For only generic case: the selection among multi processes

```
      do 3100 m = 1 , mulprc(0)
          fbsda = grcpcn(m)//'.dat'
          open(lubsdt,file=fbsda,status='unknown',
     &            form='unformatted',err=1200)
          call grcevt(-1,m,ierr)
          close(lubsdt)
 3100 continue
```

5.2 subroutine grcpar(param,iodat,iopt,ierr)

PURPOSE:
To assign a value to any external parameter or to get the value of a parameter.
PARAMETERS:

param:	`character*(*):`	Name of the parameter
iodat:	`character*8, integer*4, real*8:`	Input or output data
iopt :	`integer:`	Option

- 1 Set parameter and print
- 2 Set parameter but no-print

ierr: `integer:` Error return

- 0 Ok
- 1 Fatal error
- 10 Warning

BEWARE: The *iodat* parameter should follow precisely the type definition.

`call grcpar('amw',80.,1,ierr)` is WRONG and should be:
`call grcpar('amw',80.d0,1,ierr)`

5.2.1 Generator Parameters

Default settings are in square brackets.

- **'process'** ['`eceNuqdQ`']
  Process name
- **'energy'** [190.d0]
  Center of mass energy in GeV
- **'helicity1' ,'helicity2' ,'helicity3'**
  'helicity4' ,'helicity5' ,'helicity6' [0]
  helicity: 'left'(-1), 'average or sum' (0), 'right' (+1),
  The digit $1 \cdots 6$ refers to the particle number: For the initial electron (1), initial positron (2) the helicity zero imply taking an average of possible initial helicities. For the final state particles $(3, 4, 5, 6)$ the helicity zero imply making the sum of all helicity state. The particle numbering refers to the process name as returned by the program or as listed in section 2.4.
- **'set'** [0]
  Diagram set selection

SET	to select the Feynman diagrams to be generated:
0	all diagrams
1	W^+W^- (called CC03)
2	ZZ
3	$W^+\ell^-\overline{\nu}$ production
4	$Z\ell^+\ell^-$ or $\gamma\ell^+\ell^-$
5	$Z\nu\overline{\nu}$ or $\gamma\nu\overline{\nu}$ production
6	$\gamma - \gamma$
7	s-channel
8	t-channel
9	multi-peripheral and bremsstrahlung diagrams (6-Diagrams)
10	multi-peripheral only
11	all diagrams, but the multi-peripheral
12	ZZ or $Z\gamma$ or $\gamma\gamma$
13	SET 1 or SET 13
14	SET 4 or SET 12
15	SET 5 or SET 12

- **'isrtype'** [0]
  Type of ISR: tree (0), sf (1), qedpsi (2), qedpsif (3), qedpst(4) [1]

  sf: ISR are generated using the electron structure function technique as presented in the LEP-II yellow report (CERN 96-01 V2)

  qedpsi: Initial state radiation generating 0 to n (n $\leq$ 10) photons with the proper angular distribution. Due to the contribution of initial final interference terms in process with electrons in the final state, terms not taken into account here, cross-sections cannot be relied upon. If the final state contains 2 or more electrons the program stops. If the final state contains 1 electron and

  qedpsif: Initial and final state radiation similar to qedpsi, but should not be used for electron final state processes.

  qedpst: Initial and final state radiation similar to qedpsi for electron final state processes.

- **'jrunal'** [0]
  running alpha for the jrunal = 1 case (qedpst)

  The effect of the vacuum polarization can be included if one uses the running QED coupling in the SF method. The $ff\gamma$ coupling evolved by the renormalization group equation is given by

$$g_{ff\gamma}(-t_\pm) = g_{ff\gamma}(0)\left(1 - \frac{\alpha}{3\pi}\sum_i C_i e_i^2 \log\frac{-t_\pm}{m_i^2}\Theta(-t_\pm - m_i^2)\right)^{-1}, \qquad (1)$$

  where $\alpha = 1/137.036$ is the QED coupling at the zero momentum transfer, C_i the color factor, e_i the electric charge in unit of the e^+ charge, m_i the mass of the i-th fermion. The index i runs over all massive fermions. Only those fermions whose mass is greater than $-t_\pm$ are taken into account through the step function. The quark masses are chosen so as to match with the vacuum polarization in the `BDK` program [2]

[1]from version 2.2

[2]F.A. Berends, P.H. Daverveldt, R. Kleiss, Nucl. Phys. **B253** (1985) 412.

- **'eisr'** [0.001d0 GeV]
  Minimum photon energy for the isrtype = 1 case (sf)

- **'canon'** [1]
  canonical cuts: on (1), off (0)
  Canonical cuts are defined in the yellow report (CERN 96-01 V2)

- **'qmass' ,'lmass'** [1]
  qmass = 0 all quark mass are set to 1.d-5 GeV
  qmass = 1 quark mass are set to PDG values
  lmass = 0 all lepton mass (but the electron) are set to 1.D-5 GeV
  lmass = 1 lepton mass are set to PDG values

- **'width'** [0]
  running width width: fixed (1), running (0)
  The W width can be set "running" or "fixed". "fixed" corresponds to real tree level computation, "running" implies higher order corrections. renormalization scheme 'scheme' and the value of alpha 'alpha' cannot be selected independently. The following sets make sense:

  - 1 [Running width + g_μ scheme + alpha = 1/128] contains higher order corrections for W pair production. This is the default setup.
  - 2 [fixed width + on-shell scheme + alpha=1/137] is a strict tree level calculation.

  Gauge violation introduced by the W width is tamed using the so-called thechnique of electron current shifting as described in Phys. Lett. B349: 367-374, 1995.

- **'coulomb'** [0]
  Coulomb correction: no (0), Bardin (1), Fadin (2).
  Coulomb corrections are generated by the exchange of a virtual photon between two W. This vertex correction can be of two types:

$$\sigma_c = \sigma_o \frac{\alpha\pi}{2\bar{\beta}}[1 - \frac{2}{\pi} \arctan \frac{\mid \beta_M + \Delta \mid^2 - \bar{\beta}}{2\bar{\beta}\Im\beta_M}]$$

  where:

$$\bar{\beta} = \frac{1}{S}\sqrt{S^2 - 2S(K_+^2 + K_-^2) + (K_+^2 - K_-^2)^2}$$

$$\beta_M = \sqrt{1 - 4M^2/S}$$

$$M^2 = M_w^2 - \imath M_w \Gamma_w$$

  Bardin:

$$\Delta = \frac{\mid K_+^2 - K_-^2 \mid}{S}$$

  Fadin:

$$\Delta = 0$$

- **'qcdcr'** [0]
QCD correction: on (1), off(0)
QCD correction is a global correction applied on quark pairs production to take into account the exchange of a gluon between two quarks. The interaction vertex term is naively multiplied by:
$$1 + \alpha_s/\pi$$

- **'gluon'** [0]
Additional gluon diagrams: on (1), off(0)
These gluon diagrams are actually present in the higher order 2 fermion production where each of the 2 quarks emits a gluon converted in a quark pair. Therefore a 4 quark final state is produced whose cross-section is large, particularly if initial state radiation leads to the so-called Z return. This process should be considered as background when studying W initiated final states. It is provided here for convenience, but the user should use "JETSET"for example for more elaborated background studies.

- **'shower'** [1]
'JETSET" parton shower: on (1), off(0)

- **'scheme'** [1]
Renormalization Scheme: on-shell (0), g_μ (1)
see comment on the running width

- **'chrgconj'** [1] flag for "charge cojugate" event generation for charge conjugate processes: on (1), off(0)
Charge conjugate final states are generated randomly.

- **Anomalous coupling parameters:**

  'ang1a': g_γ^1 [1.d0]
  'ankaa': κ_γ [1.d0]
  'anlma': λ_γ [0.d0]
  'ang1z': g_z^1 [1.d0]
  'ankaz': κ_z [1.d0]
  'anlmz': λ_z [0.d0]

- **Particle mass and width:**
The second parameter is the corresponding JETSET parameter. Any change to the grc4f parameter is propagated to the jetset parameter.

'amw'	**'PMAS(24,1)'**	W mass [80.23D0]
'agw'	**'PMAS(24,2)'**	W width [2.03367D0]
'amz'	**'PMAS(23,1)'**	Z mass [91.1888D0]
'agz'	**'PMAS(23,2)'**	Z width [2.4974D0]
'ama'	**'PMAS(22,1)'**	Photon mass [0.d0]
'amg'	**'PMAS(21,1)'**	Gluon mass [0.d0]
'amh'	**'PMAS(25,1)'**	Higgs mass [10000.d0] Although all Higgs diagrams are included, the kinematics treating the additional Higgs resonances has not yet been prepared. Users should be careful when selecting small Higgs mass,

		the integration may not converge
'amel'	**'PMAS(11,1)'**	Electron mass [0.51099906D-3]
'ammu'	**'PMAS(13,1)'**	Muon mass [105.658389D-3]
'amta'	**'PMAS(15,1)'**	Tau mass [1.7771D0]
'amuq'	**'PMAS(2,1)'**	u quark mass [5.0D-3]
'amcq'	**'PMAS(4,1)'**	c quark mass [1.3D0]
'amtq'	**'PMAS(6,1)'**	t quark mass [170.0D0]
'amdq'	**'PMAS(1,1)'**	d quark mass [10.0D-3]
'amsq'	**'PMAS(3,1)'**	s quark mass [200.0D-3]
'ambq'	**'PMAS(5,1)'**	b quark mass [4.3D0]
'agh'	**'PMAS(25,2)'**	Higgs width [0.d0]

- **'alpha'** [1.0d0/128.07d0]
  α Electro-magnetic coupling constant (see comment on the 'width').
  This parameter is propagated to "JETSET" ($MSTU(101) = 0, PARU(101) = alpha$).

- **'alphas'** [0.12d0]
  α_s The strong coupling constant.
  This parameter is propagated to "JETSET" ($MSTU(111) = 0, PARU(111) = alphas$).

- **'seed'** [12345]
  Random generation seed

- **'itmx1' ,'itmx2'** [-1]
  Maximum number of iteration for the grid generation and the integration (for (-1) a 'best' value is provided from the process database)

- **'acc1' ,'acc2'** [0.1d0] [0.05d0]
  Accuracy for the grid computation and for the integration.

- **'ncall'** [-1]
  Maximum number of call to the integrand function
  (for (-1) a 'best' value is provided from the process database)

- **'mxtry'** [50]
  Maximum number of trial for the generation of 1 event

- **File logical units** [0]
  If set to 0 the file is neither written nor read.

'lubsdt'	bases data:
	Required only if event generation is foreseen on subsequent runs
'lubsrl'	bases result:
	Optional BASES Output information
'lusprl'	spring result
	Optional SPRING output information

- **'opncut'** [0.d0]
  Minimum Opening angle between two charged leptons and
  between charged lepton and quarks (degree)

- Maximum (1) and minimum (2) theta angle in degree [180.d0], [0.d0] for final state particles 3,4,5,6
  'thecut31' ,'thecut32'
  'thecut41' ,'thecut42'
  'thecut51' ,'thecut52'
  'thecut61' ,'thecut62'
- Minimum (1) and Maximum (2) energy cut on final state particle 3 4 5 6. If set to -1, kinematic limits are set (ex. engcut31=mass3, engcut32=W) [-1]
  'engcut31' ,'engcut32'
  'engcut41' ,'engcut42'
  'engcut51' ,'engcut52'
  'engcut61' ,'engcut62'
- Invariant mass minimum (1) and maximum (2) on particle couples:
  (3)(4), (5)(6), (3)(5), (4)(6), (3)(6), (4)(5). If set to -1 kinematic limits are set (ex. `invcut341`=mass3+mass4 `invcut342`= w-mass5-mass6) [-1]
  'invcut341' ,'invcut342'
  'invcut561' ,'invcut562'
  'invcut351' ,'invcut352'
  'invcut461' ,'invcut462'
  'invcut361' ,'invcut362'
  'invcut451' ,'invcut452'

5.3 subroutine grckct

User cut subroutine:

```
C----------------------------------------------------------------------
      subroutine grckct(nextrn,pe,pp,jump)
C----------------------------------------------------------------------
      implicit real* 8(a-h,o-z)
************************************************************************
*     user cut    nextrn=6  for grc4f
************************************************************************
      integer nextrn
      real*8  pe(4,nextrn), pp(nextrn,nextrn)
*
      jump = 0
*     if the point should be rejected, then set jump = 1.
      return
      end
```

6 Generic process event generation

7 Utility programs

7.1 proclist

Output the process list corresponding to a generic name:

```
proclist < list_in
```

`list_in` contains:

1. name of generic
2. number of event

Example for `list_in`:

```
CLNZUPDP
10000
```

Example for output:

```
 Input character is "CLNZUPDP" ------>   eceNuqdQ  eceNcqsQ  mcmNuqdQ
 mcmNcqsQ  tctNuqdQ  tctNcqsQ
 cross section of TREE nocut @ 172 GeV (Pb)
1  eceNuqdQ  24  2   .7148D+00    1878    1878
2  eceNcqsQ  25  2   .6959D+00    1828    3706
3  mcmNuqdQ  26  2   .5992D+00    1574    5280
4  mcmNcqsQ  27  2   .5989D+00    1574    6854
5  tctNuqdQ  28  2   .5987D+00    1573    8427
6  tctNcqsQ  29  2   .5985D+00    1573   10000
```

- Process number
- Process name
- Charge conjugate factor
- Default cross-section (pb)
- Number of event according to the relarive cross-section
- Accumulated number of event

are printed. One can change the values of cross-section. (See `util/proclist.f`)

7.2 Checkevent

This optional program reads an event file and produces a simple ntuple.

```
checkevet < checkevent.d
```

It requires the CERNLIB HBOOK package.
`checkevent.d` contains:

1. file type: direct access (1), sequential access (0)
2. Hbook output file name
3. Input Event file

4. Nb of event to read in

5. Nb of event to print out

Example:

```
0
'event.hb'     # output file
'event.out'    # event file
5000           # nb of event in
10             # nb of event to print-out
```

Comments on file type: In general, Final event files are sequential although intermediate files are direct access. Definition of the Ntuple parameters

```
*****************************************************
* NTUPLE ID=   30  ENTRIES=   1000   N-TUPLE
*****************************************************
*  Var numb  *   Name    *    Lower     *    Upper     *
*****************************************************
*      1     *    N      *  .260000E+02 *  .176000E+03 * Number of particle
*      2     *    G1     *  .000000E+00 *  .350970E+02 * Most energetic gamma
*      3     *    G2     *  .000000E+00 *  .341717E+02 * 2nd most energetic gamma
*      4     *    G3     *  .000000E+00 *  .000000E+00 * 3rd   ‘‘       ‘‘
*      5     *    G4     *  .000000E+00 *  .000000E+00 * 4th   ‘‘       ‘‘
*      6     *    KS1    *  .100000E+01 *  .140000E+02 * k(fermion3,1) Status
*      7     *    KS2    *  .100000E+01 *  .140000E+02 * k(fermion4,1)
*      8     *    KS3    *  .130000E+02 *  .140000E+02 * k(fermion5,1)
*      9     *    KS4    *  .130000E+02 *  .140000E+02 * k(fermion6,1)
*     10     *    null   *  .000000E+00 *  .000000E+00 *
*     11     *    K1     * -.110000E+02 *  .110000E+02 * k(fermion3,2) Id code
*     12     *    K2     * -.120000E+02 *  .120000E+02 * k(fermion4,2)
*     13     *    K3     * -.400000E+01 *  .400000E+01 * k(fermion5,2)
*     14     *    K4     * -.300000E+01 *  .300000E+01 * k(fermion6,2)
*     15     *    null   *  .000000E+00 *  .000000E+00 *
*     16     *    P11    * -.594526E+02 *  .581966E+02 * p(fermion3,1) px
*     17     *    P12    * -.556513E+02 *  .548355E+02 * p(fermion3,2) py
*     18     *    P13    * -.775253E+02 *  .786196E+02 * p(fermion3,3) pz
*     19     *    P14    *  .144410E+00 *  .786207E+02 * p(fermion3,4) E
*     20     *    P15    *  .510999E-03 *  .510999E-03 * p(fermion3,5) M
*     21     *    P21    * -.574834E+02 *  .542212E+02 * p(fermion4,1) px
*     22     *    P22    * -.666030E+02 *  .553676E+02 * p(fermion4,2) py
*     23     *    P23    * -.710373E+02 *  .744688E+02 * p(fermion4,3) pz
*     24     *    P24    *  .914428E+01 *  .790304E+02 * p(fermion4,4) E
*     25     *    P25    *  .000000E+00 *  .000000E+00 * p(fermion4,5) M
*     26     *    P31    * -.544322E+02 *  .745447E+02 * p(fermion5,1) px
*     27     *    P32    * -.547548E+02 *  .571485E+02 * p(fermion5,2) py
*     28     *    P33    * -.632381E+02 *  .508035E+02 * p(fermion5,3) pz
*     29     *    P34    *  .369231E+01 *  .757723E+02 * p(fermion5,4) E
*     30     *    P35    *  .500000E-02 *  .130000E+01 * p(fermion5,5) M
*     31     *    P41    * -.746067E+02 *  .668965E+02 * p(fermion6,1) px
*     32     *    P42    * -.644060E+02 *  .555779E+02 * p(fermion6,2) py
*     33     *    P43    * -.603144E+02 *  .719913E+02 * p(fermion6,3) pz
*     34     *    P44    *  .630697E+01 *  .799192E+02 * p(fermion6,4) E
```

```
*     35     *    P45    *  .100000E-01 *  .200000E+00 * p(fermion6,5) M
*     36     *    H1     * -.100000E+01 *  .100000E+01 * helicity positron
*     37     *    H2     * -.100000E+01 *  .100000E+01 * helicity electron
*     38     *    H3     * -.100000E+01 *  .100000E+01 * helicity fermion 3
*     39     *    H4     * -.100000E+01 *  .100000E+01 * helicity fermion 4
*     40     *    H5     * -.100000E+01 *  .100000E+01 * helicity fermion 5
*     41     *    H6     * -.100000E+01 *  .100000E+01 * helicity fermion 6
******************************************************
```

8 How to install the package

8.1 Setup the "grc4f" directory

1. Get the last version of the grc4f package from:
2. transfer grc4f.tar.gz
3. gunzip grc4f.tar.gz
4. tar xf grc4f.tar

The directory "grc4f" has been created: ([F] file, [D] directory)

- **Makefile:** [F]
- **amp4f:** [D]
  source for process amplitudes
- **kinem:** [D]
  source for grc4f kinematics
- **chanel:** [D]
  source for chanel library
- **basesv5.1:** [D]
  Last BASES/SPRING integrator and event generator package
- **doc:** [D]
  This documentation
- **examples:** [D]
  - **ex1:** [D]
    The simplest example
  - **ex2:** [D]
    A more complete example
  - **ex3:** [D]
    A real example with the shell script "job" for running "grc4f"
- **pro:** [D]
  source for interface code

- **util:** [D]
  source for utilities

- **bin:** [D]
  The programs `proclist` and `checkevent` (after compilation).

- **lib:** [D]
  libraries

1. Change directory to "grc4f" and edit the "Makefile" to set the Flags corresponding to your system and the correct directory path.

2. then the command "make" should prepare all libraries

3. Change dir to examples

4. Change dir to ex1, this is the first example, again check the Makefile and run "make". this should produce the grc4f executable, run it.

5. you may then go to ex2 and ex3 for more elaborated examples.

8.2 Parameters for BASES/SPRING

From version 3, one parameter of grc4f was changed. The buffer size of information of distribution of multi-dimensional integral is extented to 279936 *4 byte from 32768 *4 byte, in order to get more efficiency in event generation.

In case of TREE, QEDPS, the dimension is 7. In case of TREE, QEDPS, the dimension is 9.

dim	subregion	NCALL
7	4	< 32768
7	5	< 156250
7	6	< 559872
9	3	< 39366
9	4	< 524288

In verion 1 and 2, the subregion number is 3 for TREE/QEDPS and 2 for ISR.
Caution!

- One cannot use "`bases.data`"(related to **'lubsdt'** generated by version 2.

- "`bases.data`" has about 4.5 MB file space.