User's Manual of **CAIN**

Version 2.35 Apr.30.2003

TeXed on April 30, 2003

Contents

Chapter 1

Introduction

CAIN is a stand-alone FORTRAN Monte-Carlo code for the interaction involving high energy electron, positron, and photons. Originally, it started with the name **ABEL**[1] in 1984 for the beam-beam interaction in e^+e^- linear colliders. At that time the main concern was the beam deformation due to the Coulomb field and the synchrotron radiation (beamstrahlung). Later, the pair creation by particle-particle collision was added, and, it was renamed to **CAIN** when the interaction with laser beams (radiation by electrons/positrons and pair creation by photons in a strong laser field) was added for the γ-γ colliders.

CAIN home page is located at http://www-acc-theory.kek.jp/members/cain/

The first version **CAIN**1.1[2], which was a combined program of modified **ABEL** and a laser QED code, was limited because it could not handle the laser interaction and the e^+e^- interaction simultaneously and does not accept mixed e^+e^- beams. To overcome these problems, **CAIN**2.0 was written from scratch. It now allows any mixture of e^-, e^+, γ and lasers, and multiple-stage interactions. The input data format has been refreshed completely.

The physical objects which appear in the present version **CAIN**2.35 are particle beams, lasers, external fields and magnetic beamlines. The beams may consist of high-energy electrons, positrons and photons.

The direction of the beams is arbitrary but when the Coulomb field is to be calculated for two colliding beams, a basic assumption is that each beam must be a 'beam', *i.e.*, most particles in each beam go almost parallel. (**CAIN** assumes the two beams go opposite direction, right-going and left-going. For the case they make a large angle, you can apply **CAIN** command for Lorentz transformation so that the collision looks head-on.)

The lasers can go any direction. As external fields the present version accepts only constant fields, but since **CAIN**2.23 you can track a beam though a beamline consisting of magnets.

The physical processes that can be handled by the present version **CAIN**2.35 are

- Classical interaction (orbit deformation) due to the Coulomb field.
- Luminosity between beams (e^- e^+ γ).
- Synchrotron radiation by electrons/positrons (beamstrahlung), and pair creation by high energy photons (coherent pair creation) due to the beam field.

- Interaction of high energy photon or electron/positron beams with laser field, including the nonlinear effects of the field strength.
- Classical and quantum interactions with a constant external field.
- Incoherent e^+e^- pair creation by photons, electrons and positrons.
- Transport of charged particles through a magnetic beamline.
- In almost all interactions the polarization effects can be included.

Output data (properties of particles, luminosities, etc.) can be written in specified files at any moment of job. The graphic output is written only in the TopDrawer format. If you want other formats, you have to write a post processor by yourself.

1.1 General Structure of Input Data

In this section we briefly describe the structure of input data. **CAIN** is not intended for interactive jobs because the computing time is normally more than several minutes. Every instruction to the program is given in the input data. Two cases, a simple e^+e^- collision and a γ-γ collider, are given here as examples. For more detail look at the sections for each command and the example input data files in the directory `cain235/in`.

Consider a simple e^+e^- collision. You have first to define the two beams:

```
 BEAM  RIGHT, KIND=2, NP=10000, AN=1E10, E0=500E9, SIGT=1E-4,
      BETA=(1E-2,1E-4), EMIT=(3E-12,3E-14);
```

This defines a right-going electron (`KIND=2`) beam with the bunch population 1×10^{10}, energy 500GeV, bunch length 100μm, etc. Note that every command must end with a semicolon.

You can use variables and mathematical expressions (see Sec.2.5). For example, if you prefer normalized emittance, you may write

```
 SET  ee=500E9,  gamma=ee/Emass, emitx=3D-6/gamma,  emity=3D-8/gamma,
      betax=1E-2, betay=1E-4,
      sigx=Sqrt(emitx*betax), sigy=Sqrt(emity*betay);
 BEAM  RIGHT, KIND=2, NP=10000, AN=1E10, E0=ee, SIGT=1E-4,
      BETA=(betax,betay), EMIT=(emitx,emity);
```

`Emass` is a reserved variable and `Sqrt` is a predefined function. `sigx` and `sigy` are defined for later use. If you like millimeter instead of meter, you may say

```
 SET    mm=1E-3, sigz=0.1*mm;
 BEAM   ........ SIGT=sigz, ......;
```

Now you know how to define the positron (`KIND=3`) beam. Obviously, `BEAM LEFT, KIND=3, ...`; will do.

For calculating the beam-beam force you need to tell **CAIN** about the mesh:

```
SET Smesh=sigz/2;
BBFIELD  NX=32, NY=32, WX=8*sigx, R=sigx/sigy/2;
```

The definition of the longitudinal mesh `Smesh` may look bizzarre. This is because the same mesh is used for luminosity calculation.

For computing the e^+e^- luminosity, you have to say, for example,

```
 LUMINOSITY KIND=(2,3), W=(0,2*ee,50), WX=8*sigx, WY=8*sigy, FREP=90*150;
```

if the rep rate is 90 bunches times 150Hz. `WX` and `WY` define the mesh region (See Sec.3.11).

Now you are ready to start the collision.

```
 FLAG OFF ECHO;
 PUSH  Time=(-2.5*sigz,2.5*sigz,200);
 ENDPUSH;
```

will track the beam over the specified time range in 200 steps. It is better to turn off the echo before running. You can get the transient information (e.g., plot the beam profile during collision) by inserting commands (`PLOT`, `WRITE` etc) between `PUSH` and `ENDPUSH`. If you want the beamstrahlung, you have to insert

```
 CFQED    BEAMSTRAHLUNG;
```

before `PUSH`. After `ENDPUSH` you can plot (generate TopDrawer input file) the e^+e^- differential luminosity by

```
 PLOT LUMINOSITY, KIND=(2,3);
```

You can also plot particle distribution. For example, for plotting the photon (`KIND=1`) energy spectrum,

```
 PLOT  HIST, KIND=1, H=En/1E9, HSCALE=(0,ee/1E9,50),
        TITLE='Beamstrahlung Energy Spectrum;',
        HTITLE='EOG1  (GeV); XGX       ;';
```

`H` defines the horizontal axis (energy in units of GeV, in this example). Unfortunately, the present version creates input data for TopDrawer only.

You may want different outputs without repeating the time-consuming calculation. You can do the following. After `ENDPUSH`, store all the variables and the particle data:

```
STORE FILE='aaa';
WRITE BEAM, FILE='bbb';
```

and restore them in the input file for the next job

```
RESTORE FILE='aaa';
BEAM  FILE='bbb';
PLOT  ........;
```

γ-γ collider is more complex. Three steps, e-γ conversion of right-going electron, that of left-going electron, and γ-γ collision, are needed. You can do these steps in one job or in separate jobs using `STORE/WRITE` and `RESTORE/BEAM FILE` commands. The attached example `cain235/in/NLCggCP.i` executes the two conversions and `NLCggIP.i` the collision at the interaction point.

For the conversion you define the lasers in addition to the initial electron beam:

```
 LASER LEFT, WAVEL=laserwl, POWERD=powerd,
      TXYS=(-dcp,0,off/2,-dcp),
      E3=(0,-Sin(angle),-Cos(angle)), E1=(1,0,0),
      RAYLEIGH=(rlx,rly), SIGT=sigt, STOKES=(0,1,0) ;
```

See Sec.3.6 for the meaning of the key words. The type of laser-electron and laser-γ interactions has to be specified by LASERQED command:

```
 LASERQED  COMPTON, NPH=5, XIMAX=1.1*xi, LAMBDAMAX=1.1*lambda ;
 LASERQED  BREITW, NPH=5, XIMAX=1.1*xi, ETAMAX=1.1*eta ;
```

The PUSH-ENDPUSH loop is the same as in the e^+e^- example.

After ENDPUSH write all the particle data by WRITE BEAM, FILE=... or, if you do not want to include e-e collision, write the photon data selectively by WRITE BEAM, KIND=1, FILE=.... Then, read this file in the next job and simulate the γ-γ collision.

See Sec.2 for the basic grammer of the input data. See Sec.2.4 for a list of all the available commands.

Chapter 2

Basic Grammer of the Input Data

2.1 System of Units

MKSA is used throughout. The particle energy and momentum are eV and eV/c, respectively. An exception is the luminosity which is expressed in cm^{-2}sec^{-1}. The time (e.g., the laser pulse length, time coordinate of particles, etc.) is always expressed in units of meter by multiplying the velocity of light.

2.2 Characters

Upper and lower case alphabets are distinguished. The following characters have special use:

```
    =  ;  ,  (  [  {  )  ]  }  !  '  "
```

Also, the following characters are used in mathematical expressions:

```
    +  -  *  /  ^  =  <  >  &  $ |  .  :  (  [  {  )  ]  }
```

The command names and (almost all) keywords consist of upper case alphabets only. Variables may consist of upper/lower case alphabets, numerical characters and underscore '_'.

2.3 File Lines and Command Blocks

The input data is a collection of file lines. Upto 256 characters in a line are read in. (This limitation can be easily changed by modifying the parameter statement in the main program.)

A literal character string is defined as a string enclosed by a pair either of apostrophes `'` or of double apostrophes `"`. (See Sec.2.5.6 for more detail.) The string must close within a file line.

If a character "`!`" is encountered, the whole text after it to the end of the file line is considered as a comment, unless the "`!`" is in a literal character string.

Apart from the above two points (i.e, that a character string must close within a file line and that "`!`" is effective till the end of the file line), the concept of 'file line' is irrelevant. Therefore, for example, continuing the two file lines will give the same results,

[1] and the end of a command must explicitly stated (by semicolon "`;`") without relying on the end-of-line.

The whole text, after the comment part is eliminated, is divided into 'command blocks'. The end of a command block is indicated by a semicolon "`;`" if the "`;`" is not in a literal character string.

Each command block has the following structure:

command_name   operand`,` operand`,` $\cdots$ operand `;`

After the command_name before the first operand, there must be at least one blanck character (unless there is no operand). Operands are separated by a comma "`,`" and the number of blancks before and after "`,`" is arbitrary. (In some commands, "`,`" can be replaced by one or more blancks). Unless stated in each command description in the next section, the order of operands is arbitrary.

An operand is either a single keyword (a flag) or of the form

keyword `=` right_hand_side

A keyword is an alphanumerical string predefined for each command. The right_hand_side is just a number or an 'expression' (to be explained later) or of the form

`(` expression`,` expression`,` $\cdots$ expression`)`

The parenthesis `( )` may be replaced by `[ ]` or `{ }` if they match. In the case when all the expressions are expected to be floatng type (i.e., not character type), the right-hand-side can be replaced by an array name without subscripts. It must be a one-dimensional array and its full size is used. For example,

```
ARRAY a(2); SET a(1)=2, a(2)=3;
```
command keyword=`a;`

is equivalent to

command keyword=`(2,3);`

2.4 Commands

As stated above, each command block must start with a command name. The present version has the following commands

`ALLOCATE`	Memory allocation for big arrays. Sec.3.1.
`FLAG`	On-off flags (echo, etc.). Sec.3.2.
`SET`	Define user variables. Sec.3.3.
`ARRAY`	Allocate array variables, Sec.3.4.
`BEAM`	Define particle beams. Sec.3.5.
`LASER`	Define lasers. Sec.3.6.
`EXTERNALFIELD`	Define external (static) electromagnetic field. Sec.3.10.
`LASERQED`	Parameters for the laser-particle interaction. Sec.3.7.

[1] Here is some problem since blanck characters in a line after the last non-blanck character are ignored. For example, `SET` / `x=0` (/ is line feed) is understood as `SETx=0` even if there is a blanck character following `SET`.

`CFQED`	Parameters for the interaction between particles and constant electromagnetic field (beamstrahlung and coherent pair creation). Sec.3.8.
`BBFIELD`	Method of calculation of the beam field. Sec.3.9.
`PPINT`	Incoherent particle-particle interaction. Sec.3.12.
`LUMINOSITY`	Define what sort of luminosities to be calculated. Sec.3.11.
`LORENTZ`	Lorentz transformation. Sec.3.15.
`MAGNET`	Define a magnet for beamline transportation. Sec.3.16.
`BEAMLINE`	Define configuration of a beamline. Sec.3.17.
`MATCHING`	Optics matching of a beamline. Sec.3.19.
`BLOPTICS`	Calculate Twiss parameters of a beamline. Sec.3.18
`TRANSPORT,ENDTRANSPORT`	Loop for beam transportation along a beamline. Sec.3.20
`DRIFT`	Move particles in vaccuum or in external field. Sec.3.14.
`PUSH,ENDPUSH`	Loop of time steps. Sec.3.13.
`DO,CYCLE,EXIT,ENDDO`	Do loop. Sec.3.21.
`IF,ELSEIFELSE,ENDIF`	If block. Sec.3.22.
`WRITE,PRINT`	Print on screen or on a file. Sec.3.23.
`PLOT`	Plot using TopDrawer. Sec.3.24.
`CLEAR`	Clear data or disable commands. Sec.3.25.
`FILE`	Open/close files. Sec.3.26.
`HEADER`	Define the header for graphic outputs. Sec.3.27.
`STORE,RESTORE`	Save/recall variables and luminosity values. Sec.3.28.
`STOP`	Stop run. Sec.3.29.
`END`	End of the input file. Sec.3.30.

The command names may be shortened if not ambiguous. Therefore, `LASERQ` is equivalent to `LASERQED`. This rule applies also to the operand keywords of all commands. (But does not apply to parameter and function names.)

2.5 Expressions

In the example in Sec.1.1, the right_hand_sides of some operands are written in the form of mathematical expressions. In general, it may contain

- Literal numbers, such as `2`, `2.0`, `-3E-5`, etc.
  To indicate the exponent, any of `E,e,D,d,Q,q` may be used. Note that there is no integer expression so that `2` is identical to `2.0`.
- Literal character string enclosed by a pair of apostrophes `'` or of double apostrophes `"`.

- Arithmetic operators `+`,`-`,`*`,`/`,`^`.
- Relational operators `==`, `<`, `>`, `<=`, `>=`, `=<`, `=>`, `<>`, `><`, `/=`.
- Logical operators `&&`, `||`.
- Parenthesis: `( [ { ) ] }` . Must match.
- Parameters (variables). They are classified as scalar and array or as pre-defined and user-defined or floating and character string. There is no pre-defined array as of Cain2.3.
- Functions (pre-defined only).

The result of an expression is either a double-precision floating value or a character string. There is no integer type expression. Expressions involving character strings will be described later (Sec.2.5.6).

2.5.1 Operators

Arithmetic operators

As arithmetic operators you can use `+`, `-`, `*`, `/`, `^`. Note that power is indicated by "`^`" instead of "`**`" of FORTRAN.

Relational operators

Too many operators are defined: `==`, `<`, `>`, `<=`, `>=`, `=<`, `=>`, `<>`, `><`, `/=`. Among these, the members of each of the group (`<=`, `=<`), (`>=`,`=>`), and (`<>`, `><`, `/=`) have the same meaning.

Results of operation are either 0.0 (false) or 1.0 (true). Thus, for example, `2*(x>=y)-1` is 1 if $x \geq y$ and is -1 if $x < y$.

Note, **CAIN** does not have integer type variables so that, for example, the result of `SET x=1/5, y=(5*x>=1);` is unpredictable. Nevertheless, an integral number in floating format does not have fractional part unless the number of digits exceeds the double precision limit (about 15 digits). Therefore, `SET x=3, y=5, z=(x+2>=y);` still works as you intend.

Logical operators

In a logical expression a `&&` (or `||`) b, any number is treated as false if zero and as true if nonzero. The result of operation is either 0.0 (false) or 1.0 (true).

Priority of operators

The priority of the operators is as follows:

`^`, (`*`, `/`), (`+`, `-`), (`<`, `>`, `<=`, `>=`), (`==`, `<>`), `&&`, `||`.

The operators within a pair of braces have the same priority. In contrast to the C language, the substitution `=` is not treated as an operator. (It would cause a confusion with our command syntax keyword=operator, which is not an expression as a whole.)

2.5.2 Pre-defined parameters

There are three types of predefined parameters.

- The first type is the universal constants that never change:

`Pi`	π
`E`	$e = 2.718\ldots$
`Euler`	Euler's constant $\gamma_E = 0.577\ldots$
`Deg`	$\pi/180 = 0.0174\ldots$. You can write, e.g., `10*Deg` where the randian unit is required.
`Cvel`	Velocity of light (m/sec).
`Hbar`	Planck's constant (Joule·sec).
`Hbarc`	Planck's constant times the velocity of light (eV·m).
`Emass`	Electron mass (eV/c^2).
`Echarge`	Elementary charge (Coulomb).
`Reclass`	Classical electron radius (m).
`LambdaC`	Compton wavelength (m).
`FinStrC`	Fine structure constant.

- The second type is the parameters whose values are determined by the program. Users cannot change their values but can refer to. These variables have definte meanings only under certain situations. For example, `Time` makes sense in the `PUSH-ENDPUSH` loop, `$PrevMag` in the `TRANSPORT-ENDTRANSPORT` loop, etc. Those from `T` to `Incp` refer to each particle and, therefore, have definte meanings only in loop statements over particles (for example, in defining the axes for plots).

`Time`	Running variables for global time coordinate (m). Makes sense only inside `PUSH-ENDPUSH` loop.
`T,X,Y,S`	Running variables for particle coordinate (m).
`En,Px,Py,Ps`	Running variables for energy-momentum (eV, eV/c). The energy is `En` but not `E`.
`Sx,Sy,Ss`	Electron/positron spin. Helicity may be written approximately as `Ss*Sgn(Ps)`.
`Xi1,Xi2,Xi3`	Photon Stokes parameters ξ_1, ξ_2, ξ_3.
`Kind`	Particle species. 1,2,3 for photon, electron, positron.
`Gen`	Particle generation.
`Wgt`	Particle weight. (One macro-particle represents `Wgt` real particles.
`Incp`	1 if the particle is created by an incoherent process. Otherwise 0.
`$PName`	Particle name. 4 bytes. Normally blanck. The first character is '`T`' for test particles, '`I`' for incoherent particles. '`IBW `', '`IBH `', '`ILL `', '`IBR `' for incoherent particles created by Breit-Wheeler, Bethe-Heitler, Landau-Lifshitz, Bremsstrahlung processes, respectively. '`LOST`' for lost particles.

`Ln,Lij`	(n=0,1,2,3,4, i,j=0,1,2,3) Luminosity values used in `PLOT LUMINOSITY` command.
`W`	Center-of-mass energy used in `PLOT LUMINOSITY` command.
`Sbl`	*s*-coordinate in a beamline. Makes sense only inside `TRANSPORT-ENDTRANSPORT` loop.
`$PrevMag`	Character. Name of the previous magnet. Valid only during a `TRANSPORT-ENDTRANSPORT` loop.
`$NextMag`	Character. Name of the next magnet. Valid only during a `TRANSPORT-ENDTRANSPORT` loop.
`$InFilePath`	Full path of the input file directory. (includes the last character '/' (UNIX) or '\' (Windows)).
`$InFileName`	Name of the input file (without path, with extension).

- The third type is those whose names are predefined with default values and which the user can change (by `SET` command) such as

`MsgFile`	File reference number for echo, error messages, and default destination of `PRINT` command. (default=6)[2]
`OutFile`	File reference number for voluminous outputs. The default destination of `WRITE` command. (default=12)
`OutFile2`	Other print output. Not used. (default=12)
`TDFile`	TopDrawer file number. (default=8)
`MsgLevel`	Message level. (default=0, i.e., error messages only)
`Rand`	Random number seed. Positive odd integer other than 1, default=3. You can reset random number at any time.
`Debug`	Debug parameter for the programmer. If you set `Debug`≥2, call and return from major subroutines are announced. (default=0)
`Smesh`	Longitudinal mesh size (m) for the calculation of beam-beam field, luminosity, etc. No default value.

2.5.3 User-defined parameters

These are those defined by `SET` command. (see Sec.3.3) Upto 16 characters consisting of upper/lower case alphabets, numericals, and underscore '_'. The first character must not be a numerical. The first character of character type variables must be '$' (and no '$' in the body).

2.5.4 Predefined functions

There are following basic math functions.
`Int,Nint,Sgn,Step,Abs,Frac,Sqrt,Exp,Log,Log10,`

[2]The input file number is set to 5. If you want to change it, see the variable `RDFL` in the file 'cain235/src/initlz.f'.

```
Cos,Sin,Tan, ArcSin,ArcCos,ArcTan,
Cosh,Sinh,Tanh, ArcCosh,ArcSinh,ArcTanh, Gamma,
Mod, Atan2, Min, Max
```

Defintions are the same as in standard FORTRAN except `Sgn` and `Step`:

$$\texttt{Sgn}(x) = \begin{cases} 1 & \text{for } x > 0 \\ 0 & \text{for } x = 0 \\ -1 & \text{for } x < 0 \end{cases} \qquad \texttt{Step}(x) = \begin{cases} 1 & \text{for } x \geq 0 \\ 0 & \text{for } x = 0 \end{cases}$$

Enclose the argument by `( )` or `[ ]` or `{ }`. Separate arguments by "`,`" if there are more than one argument (`Mod`, `Atan2`, `Min`, `Max`). (Number of arguments for `Min` and `Max` is arbitrary.)
In addition to the above functions of standard type there are functions of other type, which are defined for **CAIN**. See the next subsection Sec.2.6.

2.5.5 Arrays

You can allocate (or deallocate) arrays by using the command `ARRAY` and set their values by `SET` command. The rule about the array name is the same as that of user-defined parameters. The subscripts are delimited by commas and enclosed by a pair of parenthesis `( )` or `[ ]` or `{ }`(must match). For example,

```
ARRAY  a(20,0:10);
SET    a(3,4)=5.0,  x=a(3,4);
ARRAY  FREE a;
```

When the lower bound of a subscript is not specified in `ARRAY` command, 1 is assumed as in FORTRAN. Total number of arrays and the maximum rank is limited (but reasonably large number) by FORTRAN parameter statement but the size of each array is arbitrary as long as your computer memory allows. See Sec.3.4 for more detail.

2.5.6 Character expression

A literal character string is defined as a string enclosed by a pair either of apostrophes `'` or of double apostrophes `"`. The string must close within a file line.[3] When you need a long string, you can use concatenation like `'abc' + 'xyz'` to be explained below.

Within a string enclosed by `'` (`"`) you can use `"` (`'`) as a normal character. The FORTRAN rule that two successive `'` or `"` are recognized as a single `'` or `"` is still valid but not reccommended (there can be bugs). Even if you have both `'` and `"` in a string, you can write like `"'"+'"'`.

Character expressions have been introduced since **CAIN**2.3. All expressions can be classified into two types, floating and character, according to the results.

[3]When a file line is read from a file, blanck spaces to the end of the line cause a problem. There is no way in standard FORTRAN to distinguish between blanck spaces existing in the file and those added when read in.

A variable with the first character `$` (up to 16 characters including the `$`, no `$` in the body) is treated as a character string variable. For example,

```
    SET $a = 'abc', $b=$a + 'xyz';
```

will define `$b` as a string containing `'abcxyz'`. Basically, when an operand of a command is of the form keyword=`'`something`'` such as file names, you can use a general form of character expression including character variables for the right-hand-side.[4] For example,

```
   SET $fn='abc', n=3;  FILE OPEN, UNIT=20, NAME=$fn+$ItoA(n)+'.txt';
```

opens a file with the name `'abc3.txt'`. (See below for `$ItoA`.)

There are various limits for the size of character stack (e.g., total number of variable names, total stored number of characters, total number of characters in one expression, etc). They cannot be changed by the `ALLOCATE` command, basically because FORTRAN90 does not allow dynamic allocation of a character string of dynamically determined length. However, the prepared sizes are large enough for normal uses.

The only possible operations involving character strings are

- Concatenation by `+`. The result is a character string.
- Multiplication by a positive integer like, e.g., `2*$a` (or `$a*2`) which is equivalent to `$a+$a`. The result is a character string.
- Relational operation like `$a==$b`. The result is a floating number either 0.0 or 1.0. The result of `$a>$b` may depend on the platform (lexical order or ascii code order). Note that due to the standard FORTRAN rule the result of `$a==$b` is true (1.0) when `$a='abc'` and `$b='abc '`.

You can introduce arrays of character strings in the same way as floating numbers, e.g., by

```
    ARRAY  $a(3,0:5);
```

This only defines a pointer. Strings are actually allocated when defined by `SET` command. The elements of an array may have different lengths.

There are a few functions related to character strings. Obviously, when the first character of the function name is `$`, it returns a character string, otherwise a floating number.

`Strlen`	Length of a character string, e.g., `SET n=Strlen($a);`
`AtoF`	Convert a character string into a floating number, e.g., `SET $a='1e10', x = AtoF($a);`

[4] However, when this manual says 'apostrophes can be omitted', such as the case of magnet names, you cannot use the general form. You have to use the name alone or the name enclosed by apostrophes.

`$FtoA`	Convert a floating number into a character string. A format must be specified like `SET $a=$FtoA(5.3,'(F4.2)');` which will define a string `$a='5.30'`. `( )` will be added when the format string is not enclosed by `( )`.

`$ItoA`	Convert a floating number into a character string after operating `Nint`. For example,
	`SET $a=$ItoA('3.2');`
	will create a string `$a='3'`. You can optionally specify the format like
	`SET $a=$ItoA('4.2','(I3.3)');`
	which results in `$a='004'`. (Consult your FORTRAN manual.) `( )` will be added when the format string is not enclosed by `( )`.

`$Substr`	Substring. `$Substr($a,`n_1`,`n_2`)` is the substring of `$a` from n_1-th character (start from 1) to n_2-th character. If n_2 is omitted, the end of `$a` is used. A null string is returned if $n_2 < n_1$.
`Strstr`	`Strstr($a,$b)` searches for the first occurence of the string `$b` within the string `$a` and return the position of the first character (start from 1) if found. Return 0.0 if not found or if illegal (zero length of `$b` etc).
`$ToUpper`	Get a string with all lower case characters converted to upper case. `SET $b=$ToUpper($a);`
`$ToLower`	Get a string with all upper case characters converted to lower case.

2.6 CAIN functions

In addition to the predefined functions of general use, such as `Sin` and `Cos`, there are other special functions intrinsic to **CAIN**. They are:

Beam statistics functions	Average/rms quantities of the beam such as `SigX`.
Test particle functions	Retrieves parameters of individual test particles.
Beamline functions	Twiss parameters, etc.
Luminosity-related functions	Retrieves luminosity
Laser-related functions	Local laser intensity, etc.
Special functions	Such as Bessel functions.

2.6.1 Beam statistics functions

The number of particles, that of macro particles, the average coordinates/energy-momentum and their r.m.s. values of the beam at the given moment are retrieved by

```
NParticle, NMacro,
AvrT, AvrX, AvrY, AvrS, SigT, SigX, SigY, SigS,
AvrEn, AvrPx, AvrPy, AvrPs, SigEn, SigPx, SigPy, SigPs,
```

BeamMatrix

The calling sequence is common to these functions except `BeamMatrix`. Let us take `SigX` as an example.

`SigX(`j`,`k`)` (j= 1 or 2 or 3, k= 1 or 2 or 3)

returns the horizontal r.m.s. size of right-going (j=1) or left-going (j=2) or both (j=3) of the photon (k=1) or electron (k=2) or positron (k=3) beam. **The particles created by incoherent processes are excluded**. (See below for how to include them.)

There can be one more argument,[5] which must be enclosed by a pair of apostrophes (a character expression in general), like

`SigX(`j`,`k`,'`f`')`

where f is a logical expression for selecting particles. It may contain variables of individual particles (such as `En`, `X`, etc). Then, the particles that make f true (i.e., $\neq 0$) are selected. For example,

`SigX(1,2,'En>1e9 && En<2e9')`

will select right-going electrons with energy between 1 and 2 GeV. (See Sec.3.31 for more detail.) The reason that f must be enclosed by apostrophes is that f must not be evaluated immediately but is to be evaluated later individually for each particle.

When the particle selection argument is given, the incoherent particles are included by default. If you want to exclude them, you should say, e.g.,

`SigX(1,2,'En>1e9 && En<2e9 && Incp==0')`

When you include incoherent particles only, you should of course say `Incp==1`.

`BeamMatrix` requires two more arguments

`BeamMatrix(`a`,`b`,`j`,`k`,'`f`')` ($1 \leq a,b \leq 8$)

The returned value is the average of $x_a x_b$ where x_a=(`T`,`X`,`Y`,`S`,`En`,`Px`,`Py`,`Ps`) for a=1 to 8. (In units of m and eV or eV/c.)

2.6.2 Test particle functions

The coordinates and the energy momentum of the test particles can be retrieved by the functions

TestT, TestX, TestY, TestS, TestEn, TestPx, TestPy, TestPs

The calling sequence is, for example, `TestX('name')` or `TestX(`n`)`, where `'name'` is the character string for the particle name and n is an expression representing an integer $-99 \leq n \leq 999$. (See Sec.3.5 for the test particle name.)

2.6.3 Beamline functions

The functions related to the beamline optics such as β and η functions are retrieved by

Beta, Alpha, Eta, Etaprime, Nu

(`Nu` is the phase advance $/(2\pi)$ from the beamline entrance.) Prior to use, you must compute the optics by `BLOPTICS` command. The calling sequence of these functions are the same.

[5]The meaning of this argument has changed since **CAIN**2.3. It used to be selecting the range of `S` coordinate but has been replaced by a more powerful one.

`Beta(`j`,` mag `,` 'bl_name'`)`
where

j	1 or 2 for x or y, respectively.
mag	Either integer or character expression. If it is an integer $n > 0$, the **exit** of the n-th magnet in the beamline is implied. (Beamline entrance if $n \leq 0$ and beamline exit if n is equal to or larger than the number of magnets in the beamline.) If mag is a character expression, it is identified as a magnet name. If there are more one magnets with the same name, you can add the occurence number after a dot. For example, the exit of the third `QF` is `'QF.3'`. You can of course write as `'QF.'+$ItoA(3)`. If omitted, the first occurence is assumed. If you forget apostrophes and write `Beta(1,QF,'blname')`, you will get an error message like '`Variable QF not found`'.
bl_name	Beamline name. Must be enclosed by apostrophes. (character expression in general)

You can omit the third argument if in a `TRANSPORT-ENDTRANSPORT` loop. The default is the current beam line. (Even in this case you have to call `BLOPTICS` command prior to `TRANSPORT`. Note that Twiss parameters are not needed for particle tracking.) You can also omit the second argument, which means the current position.

You can omit the third argument during optics matching (i.e., in the matching condition), but cannot omit the second argument in this case.

These functions return zero when an error occurs (such as `BLOPTICS` not called, the beamline not existing, illegal number after the dot, etc.).

2.6.4 Luminosity-related function

There are functions related to the luminosity:
`Lum, LumH, LumP`
`LumW, LumWbin, LumWbinEdge, LumWH, LumWP`
`LumEE, LumEEbin, LumEEbinEdge, LumEEH, LumEEP`
See Sec.3.11 for definitions for these functions.

2.6.5 Laser-related function

`LaserIntensity(`t`,`x`,`y`,`z`,`n`)` Get laser intensity in Watt/m^2 at the space-time point (t,x,y,z) (world coordinate, not laser coordinate) for laser #n. (n can be omitted if n=1.)

`LaserRange(`i`,`j`,`n`)` Get the range where the laser field is non-zero in laser coordinate for laser #n. (n can be omitted if n=1.)
i=1: minimum, i=2: maximum,
j=0: $\tau - \zeta$, j=1: ξ, j=2: η, j=3: ζ
See Sec.5.8.1 for the laser coordinate (τ, ξ, η, ζ).

2.6.6 Special functions

Bessel function J_n. (n must be an integer.)

`BesJ(`n`,`x`)`	Bessel function $J_n(x)$. n must be an integer.
`DBesJ(`n`,`x`)`	Derivative of Bessel function, $J'_n(x)$. n must be an integer.

Modified Bessel function K_ν and its integral. In all the following functions, the last argument k must be 1 or 2. When $k = 2$, the output is the function multiplied by e^x. The last argument may be omitted (equivalent to $k = 1$.)

`BesK(`ν`,`x`,`k`)`	Modified Bessel function $K_\nu(x)$. $(x > 0)$
`DBesK(`ν`,`x`,`k`)`	Derivative of the modified Bessel function $K'_\nu(x)$. $(x > 0)$
`BesK13(`x`,`k`)`	Modified Bessel function $K_{1/3}(x)$. $(x > 0)$
`BesK23(`x`,`k`)`	Modified Bessel function $K_{2/3}(x)$. $(x > 0)$
`BesKi13(`x`,`k`)`	Integral of Modified Bessel function, $Ki_{1/3}$. $(x > 0)$ See eq.(5.142) for the definition of Ki.
`BesKi53(`x`,`k`)`	Integral of Modified Bessel function, $Ki_{5/3}$. $(x > 0)$

Functions for beamstrahlung and coherent pair creation.

`FuncBS(`x`,`Υ`)`	Beamstrahlung function F_{00} defined in eq.(5.140). x $(0 < x < 1)$ is the photon energy in units of the initial electron energy. $\Upsilon > 0$.
`FuncCP(`x`,`χ`)`	Spectrum function F_{CP} of coherent pair creation defined in eq.(5.156). x $(0 < x < 1)$ is the positron energy in units of the initial photon energy. $\chi > 0$.
`IntFCP(`χ`,`k`)`	Integral of `FuncCP(`x`,`χ`)` over $0 < x < 1$. The total rate of coherent pair creation is given by multiplying by $\alpha m^2/(\sqrt{3}\pi E_\gamma)$. (See Sec.5.10). k must be 1 or 2 (can be omitted if 1). If $k = 2$, the function is multiplied by $\exp(8/3/\chi)$.

2.7 Meta-expression

Some of the **CAIN** functions, e.g., the beam statistics functions, accept an expression enclosed by apostrophes as an argument. For example, as stated above, `SigX(1,2,'En>1e9')` retrieves σ_x of right-going electrons with energy above 1GeV. If this expression is written as `SigX(1,2,En>1e9)`, although gramatically incorrect for `SigX`, the expression `En>1e9` would be evaluated in place and give one single number 0.0 or 1.0. On the other hand the argument `'En>1e9'` in `SigX(1,2,'En>1e9')` does not mean one single value but is to be evaluated for each particle repeatedly. This kind of expression may be called 'meta-expression'. Note that `SigX(1,2,'En>1e9')` can also be written as `SigX(1,2,$a)` if `$a` is already defined by `SET $a='En>1e9';`.

There are other occurences of meta-expressions, although not enclosed by apostrophes, such as the `SELECT` operand of many commands, `H` and `V` operands of `PLOT` command, etc. These need not be enclosed by apostrophes because there is no fear of confusion.

The concept of 'meta-expression' is important in particular in optics matching. When you define a magnet, you have to distinguish between a parameter that is constant during matching and one that changes when the variables change. The latter must be written as a meta-expression (i.e., a character variable or as a character string enclosed by apostrophes). For example, `MAGNET 'QF', L=s, K1=x/2;` defines a magnet of length `s` and strength `x` using the current value of `s` and `x`, but `MAGNET 'QF', L='s', K1='x/2';` will change when `s` or `x` changes later.

CAIN evaluates expressions by a sort of an interpretator. Since this is very slow, a sort of compiler and loader is employed when a meta expression is to be evaluated many times (e.g., repeated over particles). A problem happens when a meta-expression contains another meta expression, which may happen, for example, if you use `SigX` in `H` operand of `PLOT` command. This is solved by FORTRAN recursive call of the loader since **CAIN**2.3 (FORTRAN90 required).

However, this is still very much time consuming because the compiler is invoked many times for the inner meta-expression in the present algorithm. Therefore, you should avoid recursive call of meta-expressions. (There can also be bugs related to the recursive call.)

2.8 External Files

Files are identified by unit number and/or file name.

Standard I/O files

The files used for standard outputs are refered to only by unit numbers defined by the variables `MsgFile`, `OutFile`, `OutFile2`, and `TDFile`. These unit numbers should be ascociated to particular file names before **CAIN** run, if needed. These files are not closed unless you do so by `FILE CLOSE` command.

On the other hand the input file unit number is not asigned to a **CAIN** variable but is fixed to 5. If you want to change it, you have to change the FORTRAN varible `RDFL` in '`cain235/src/initlz.f`' and compile this file.

Other output files

Some commands such as `WRITE` and `BEAM` accept I/O files specifed as `FILE=`f_n|`'`file_name`'`. The right-hand-side is evaluated as a general expression. If it is of floating type, it is identified as a file unit number $f_n > 0$. If it is of character type, it is identified as a file name. Either full path or relative path can be used but note that **CAIN** is run in the directory `cain/exec` (UNIX version) or in the directory where the input file is located (Windows version).

When you specify the unit number `FILE=`f_n, f_n must be associated to an actual file name in advance before **CAIN** run or by the command `FILE OPEN`. In this case the file is not closed till the end of **CAIN** run unless you explicitly close it by `FILE CLOSE` command. Therefore, when you use the same unit number in the same run again, the new data will be appended (`APPEND` operand is not needed).

When you specify a file name `FILE='`file_name`'` in a command, the file is opened with a temporary unit number. The file is closed at the end of execution of the command. If you want to keep writing onto the same file, you have to use `APPEND` operand except in the first call. Otherwise the file will be overwritten. (Actually, in such a case, better to use the form `FILE=`f_n.)

Chapter 3

Commands

A command, in general, has the following structure:

command_name   op$_1$`,` op$_2$`,` ... `,` op$_n$ `;`

A command_name is a string consisting of upper-case roman letters only. There must be one or more than one blanck characters after a command_name before the first operand.

'op$_j$' is an operand having either one of the following forms:

(a) kwd
(b) expr
(c) kwd `=` expr
(d) kwd `= (` expr `,` expr `, ... )`

Here, 'kwd' is a keywaod, i.e., a string consisting of upper-case roman letters only, which is predefined for each command. 'expr' is a mathematical expression described in Sec.2.5.

An operand of the form (a) is a flag-type operand.

The form (b) is exceptional. It is used only when printing the value of a variable.

The right-hand-side of type(c) can be a character string for some operands.

The right-hand-side of type(d) can also be an array name. If, for example, `a` is an array of length 2, kwd `= a` is equivalent to kwd `= (a(1),a(2))`.

In some commands, the first operand must be a positional operand of the flag-type. (For example, `LASERQED` command must be either `LASERQED COMPTON` or `LASERQED BREITWHEELER`.) In such a case, the "`,`" after the keyword may be omitted. (There is no ambiguity because keywords do not contain blanck characters in contrast to expressions.) `FLAG` command is special in that all the commas may be omitted because all the operands are type(a).

The command names and the keywords can be shortened so long as unambiguos. For example `LASERQ` is equivalent to `LASERQED` since the former can distinguish from `LASER`.

Now, let us describe the each command in detail. When describing the command formats in this manual, the type-faced characters are those to be typed in the input data as they are. (The variable names in the FORTRAN source also appear in type-face.) The items embraced by square brackets [ ] may be omitted in some cases and the vertical stroke "|" indicates an exclusive choice of one of the items. Thus, [`A`|`B`] means to choose either one of `A` or `B` or to omit both. Note that [ ] and `[ ]` are different.

The dagger † indicates that the operands to the left of it are positional operands.

The quantities printed in math-font in command syntax can be expressions.

3.1 ALLOCATE

Allocate memory for some of the arrays. Dynamic memory allocation has been used since **CAIN**2.2. Some of the big arrays are allocated near the beginning of run so that you need not re-compile the program when large memory is needed. (Dynamic allocation requires FORTRAN90 but is absolutely needed for Windows version because only the binary is distributed.)

- This command must not be preceeded by other commands except for `HEADER` and `SET` commands.
- If this command is not invoked, **CAIN** allocates the arrays using the default values when the first command other than the above two commands is encountered.
- This command can appear only once. Therefore, for example,
  `ALLOCATE MP=100000; ALLOCATE MVPH=10000;`
  is illegal.

Syntax:

`ALLOCATE` [`MP=` m_p ,] [`MVPH=` m_{vph} ,] [`MMAG=` m_{mag} ,] [`MBEAMLINE=` m_{bl} ,] [`MBBXY=` m_{bb} ,] [`MLUMMESH=` $m_{\mathcal{L}}$,] ;

m_p	Maximum number of macro particles. Default is 10^5.
m_{vph}	Maximum number of virtual photons in a time step in a longutudinal slice. Default is $m_p/10$.
m_{mag}	Maximum number of magnets to be defined in `MAGNET` command. Default is 200.
m_{bl}	Maximum number of beamlines to be defined in `BEAMLINE` command. Default is 50.
m_{bb}	Maximum number of bins (for each of x and y) for the calculation of the beam-beam force. Choose a power of 2. Default is 128.
$m_{\mathcal{L}}$	Number of bins (for each of x and y) for the luminosity calculation. Choose a power of 2. Default is 128.

3.2 FLAG

Set flag. example:

```
          FLAG  ON ECHO OFF SPIN ;
```

The keywords `ON` and `OFF` act until the opposite one appears. `ON` is the default after `FLAG`.
Existing flags

`ECHO`	input data echo (default=`ON`)

`SPIN`	include spin calculation (default=`ON`) (Sorry, spin calculation cannot be avoided consistently in the present version.)

3.3 SET

Defines parameters.
Syntax:

`SET` [p = a] [, p = a] $\cdots$;

p	New or existing parameter name or an element of an existing array. The name can consists of upto 16 characters, upper/lower case alphabets, numericals, or underscore '`_`'. The first character must not be a numerical. It must be `$` for character type variables/arrays. Unchangeable predefined parameters (`Pi`, `Time`, etc) and the predefined function names (`Sin`, etc) have to be avoided. All the predefined parameter names and function names start with an uppercase letter. Therefore, a user parameter starting with a lower case alphabet will never hit the predefined ones.
a	An expression. See Sec.2.5. The type (floating or character) must match with the left-hand-side.

3.4 ARRAY

Allocate or deallocate arrays.
Syntax:

`ARRAY` a($[l_1$:$]n_1$, $[l_2$:$]n_2$, $\cdots$, $[l_m$:$]n_m$) [=v], $\cdots$;

a	Array name to be allocated. Obeys the same rule as user-defined parameter (upto 16 characters). If the array already exists, it is once freed and then re-allocated. The first character must be `$` for character type arrays.
l_j,n_j	Define the lower and upper bounds of the subscripts. If l_j is omitted, the subscripts start at 1 as in FORTRAN.
v	All the elements of a floating array are initialized by the value v. (Default=0.0) This is ignored for character arrays (They are initialized by null strings.)

The syntax to deallocate arrays is
Syntax:

`ARRAY` `FREE,` a_1 [, a_2] $\cdots$;

This is not needed unless you want many arrays or unless your computer memory is very limited. Those left unfreed are deallocated automatically at the end of the **CAIN** run.

3.5 BEAM

Defines a beam. (Append particles to the existing list.) There are two ways to create a beam, one by specifying the Twiss parameters, etc, and the other by reading data from a file. See Sec.5.1 for the coordinate system.

3.5.1 Definition by Twiss parameters

Note that the beam is defined on a plane s=constant (race-goal picture), rather than on the t=constant plane (snap shot picture). Thus, e.g., the bunch length is a spread in t (although in units of meter) rather than in s.
Syntax:

BEAM RIGHT|LEFT, KIND=k, AN=N, NP=N_p, E0=E_0,
[TXYS=(t,x,y,s),] BETA=(β_x,β_y), [ALPHA=(α_x,α_y),] [EMIT=(ϵ_x,ϵ_y),]
[SIGT=σ_t,] [SIGE=σ_ε,] [GCUT=(n_x,n_y),] [GCUTT=n_t,] [GCUTE=n_ε,]
[GAUSSWEIGHT=i_g,] [ELLIPTIC,] [TUNIFORM,] [EUNIFORM,]
[SLOPE=(θ_x,θ_y),] [CRAB=(ψ_x,ψ_y),] [ETA=(η_x,η_y),]
[ETAPRIME=(η'_x,η'_y),] [ESLOPE=$d\varepsilon/dt$,] [XYROLL=ϕ_{xy},]
[DALPHADE=($d\alpha_x/d\varepsilon$,$d\alpha_y/d\varepsilon$),] [SPIN=(ζ_x,ζ_y,ζ_s),] ;

RIGHT|LEFT Specify whether the beam is right-going or left-going.

k Particle species. 1 for photon, 2 for electron, 3 for positron. If you cannot remember these codes, you can do

```
SET photon=1, electron=2, positron=3 ;
BEAM RIGHT, KIND=electron ...
```

N Number of real particles.

N_p Number of macro-particles.

E_0 Beam energy. (eV)

t, x, y, s Location of the reference point and the time when the beam center comes there. In units of meter. This is the point where the Twiss parameters are to be defined. Default=(0,0,0,0).

β_x, β_y Beta functions (m).

α_x, α_y Alpha functions. Default=(0,0). The sign of α is positive when the beam is going to be focused, regardless the beam is right-going or left-going.

ϵ_x, ϵ_y R.m.s geometric emittance (rad·m). Deafault=(0,0).

σ_t R.m.s. bunch length (m). Default=0.

σ_ε Relative r.m.s. energy spread. Default=0.

n_x, n_y, n_t, n_ε Gaussian tail cutoff in units of corresponding sigmas. The default values are 3. for n_x and n_y, n_t and n_ε. (For transverse variables the cut off is done in the action variable, which means $J_i/\epsilon_i \le n_i^2/2$ (i=x,y).)

i_g 0 or 1. There is a subtle problem on how to take into account the Gaussian cut off (n_x,n_y,n_t) in the macro-particle weight. **CAIN** throws away the random numbers outside this range and generates exactly N_p macro-particles. This means some fraction outside the region is moved inside. Therefore, if the simple weight N/N_p is assigned to macroparticles ($i_g = 1$), the effective particle density would become slightly larger than the physical value, although the sum of the weight is equal to N. If one is interested in the quantities related to the density (such as luminosities), this would cause an overestimation.
When $i_g = 0$ (default), a correction factor is multiplied to the weight such that the real particle density becomes correct. In this case, the sum of the macro-particle weights is less than N. (When the default n's are adopted, for example, the correction of the weight amounts to $\sim$3.4%.) In most cases, $i_g = 0$ will be better.

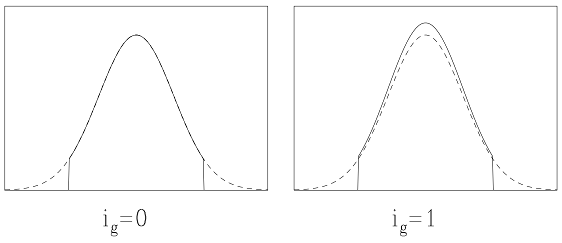


Figure 3.1: Physical charge density (dashed curve) and the simulated density (solid) for i_g=0 and 1

`ELLIPTIC` Uniform transverse distribution. (Default is Gaussian.) (x, y) distribution is a uniform ellips with radii $(2\sigma_x, 2\sigma_y)$, where $\sigma_j = \sqrt{\epsilon_j\beta_j}$ (j=x,y). In this case the beam is parallel, in spite the finite emittances are specified. The emittance and beta are only used to define $\sigma_{x,y}$. `ALPHA` and `GCUT` are not used.

`TUNIFORM` Uniform t-distribution. (Default is Gaussian.) The full length is $2\sqrt{3}\sigma_t$. `GCUTT` is not used.

`EUNIFORM` Uniform E-distribution. (Default is Gaussian.) The full relative energy spread is $2\sqrt{3}\sigma_\varepsilon$. `GCUTE` is not used.

θ_x, θ_y Angle offset (radian). The right and left-going beams have the same sign of slope when there is a crossing angle. Default=(0,0).

ψ_x, ψ_y Crab angle $\partial x(y)/\partial t$. (radian). Positive when the bunch tail has larger x (y).
When the full crossing angle in the horizontal plane is ϕ_{cross} and this is to be compensated by the crab angle, the `SLOPE` and `CRAB` parameters should be `SLOPE=`$\phi_{cross}/2$`,` `CRAB=`$\phi_{cross}/2$`,` for both right-going and left-going beams.
If you are not confident, after beam definition try, for example,

```
 DRIFT T=t0-dt ;
 PLOT SCAT, H=S, V=X,
   HSCALE=(smin,smax), VSCALE=(xmin,xmax),
   HTITLE='S(m)', VTITLE='X(m)' ;
 DRIFT T=t0 ;
 PLOT SCAT, NONEWPAGE, H=S, V=X,
   HSCALE=(smin,smax), VSCALE=(xmin,xmax),
 DRIFT T=t0+dt ;
 PLOT SCAT, NONEWPAGE, H=S, V=X,
   HSCALE=(smin,smax), VSCALE=(xmin,xmax),
```

with appropriate definitions of `t0`, `smin` etc. The `DRIFT` command transports the beam to the plane t=constant (snap shot). `NONEWPAGE` operand suppresses page break so that the (s,x) profiles at different times appear on the same page.

η_x, η_y	Eta function (m).
η'_x, η'_y	Derivative of eta function.
$d\varepsilon/dt$	Coherent energy slope from bunch head to tail (1/m).
ϕ_{xy}	Roll angle of the beam in the x-y plane. (radian)
$d\alpha_x/d\varepsilon$, $d\alpha_y/d\varepsilon$	Energy dependence of α_{xy}. This defines the energy dependence of the focal point. The focal point in the $x(y)$ plane for particles with relative energy deviation ε is given by $\varepsilon\beta_{xy}\times d\alpha_{x(y)}/d\varepsilon$ (positive if focused beyond the normal focal point). This operand together with the coherent energy slope $d\varepsilon/dt$ can be used for the travelling focus scheme.
ζ_x, ζ_y, ζ_s	Polarization vector. Default=(0,0,0). Note the sign of ζ_s for left-going particles. In the case of photon beams, these are the Stokes parameter (ξ_1, ξ_2, ξ_3). The basis vector of the Stokes parameter is $(\boldsymbol{e}^{(1)}, \boldsymbol{e}^{(2)}, \boldsymbol{e}^{(3)})$ where $\boldsymbol{e}^{(3)}$ is the unit vector along the particle momentum, $\boldsymbol{e}^{(1)}$ is the unit vector along $\boldsymbol{e}_x - \boldsymbol{e}^{(3)}(\boldsymbol{e}^{(3)}\cdot\boldsymbol{e}_x)$, and $\boldsymbol{e}^{(2)} = \boldsymbol{e}^{(3)}\times\boldsymbol{e}^{(1)}$.

See Sec.5.3 for rigorous definitions.

3.5.2 Read particle data from a file

Standard format

When you read a file of **CAIN**-defined format (standard format, MATHEMATICA format, FORTRAN NAMELIST), the syntax is

Syntax:

`BEAM` `FILE=`f_n|'file_name', [`N=`N_p`,`] [`NAMELIST,`] ;

f_n,file_name	Unit number or name of an existing file. For the differece between the two forms, See Sec.2.8.
N_p	Maximum number of macro-particles to be read in from file. If 0, non-active. Default=0.
`NAMELIST`	FORTRAN NAMELIST format. Othewise the standard format.

Reading file stops when one of the following conditions are satisfied.

- N_p is reached (when $N_p > 0$).
- A file line found whose first three characters are 'END' in the case of the standard format. Or, `END=.TRUE.` found in the case of the NAMELIST format.
- End_of_file detected.

In the case of the standard format, the file is assumed to be created by the following FORTRAN statement.

```
     WRITE(*,'(I2,I6,1X,A4,1P12D20.12)') KIND,GEN,NAME,WGT,
    1      (TXYS(I),I=0,3),(EP(I),I=0,3),(SPIN(I),I=1,3)
```

Here, `NAME` is blanck unless the particle is a test particle or a lost particle or an incoherent-pair particle, `WGT` is the number of real particles expressed by one macro-particle and `GEN` is an integer expressing the generation (1 for the initial particles, 2 for secondaries, etc.). `SPIN` is the polarization vector for electron/positron and the Stokes parameter for photons.

The file can also be MATHEMATICA style (automatically detected). The format string is `('{',I1,',',I5,',',A4,12(',',1PD19.12),'},')`

In the case of the NAMELIST format, the namelist `BEAMIN` must be inserted for each particle.

```
 &BEAMIN
   KIND=2, GEN=1, PNAME='    ', WGT=0.0, TXYS=0.0, 0.0, 0.0, 0.0,
   EP=0.0, 0.0, 0.0, 1.0,  SPIN=0.0, 0.0, 0.0,
   END=.FALSE., SKIP=.FALSE.
 &END
```

Here, the r.h.s. show the number of data, the data type, and the default. The last component of `EP`, i.e., P_s, must not be zero. All the particles must be either right- or left-going. (Actually the particle energy is calculated from $\sqrt{\boldsymbol{p}^2 + m^2}$. The input data is not used.)

If the first character of `PNAME` is `T`, the particle is treated as a test particle. (The test particle name must be unique.) `PNAME` should be blanck for normal particles.

If `SKIP=.TRUE.`, the present data is omitted. If `END=.TRUE.`, the present data and all the following data are ignored. Comments in NAMELIST statements follow the local rule on the platform.

To modify the file data (shift of origin, rotation, etc) can be done to some extent by using the command `LORENTZ`.

User-defined format

You can also read a file of user-defined namelist-like format. The generic form of the file data that **CAIN** can undestand is

begin_flag   keyword=value,   keyword=value, . . .,   end_flag

for each particle. You have to list up these keywords in the command and to give formulas to convert them into **CAIN** variables.

The syntax of the command is

Syntax:

```
BEAM    FILE=fn|'file_name',   [N=Np,]  USERDEFINED,   [BEGIN='b-str',]
  [END='e-str',]   [TERMINATE='t-str',]   [COMMENT='c',]
  KEYWORD=(kw1,kw2,...),   CONVERSION=(x1=f1,x2=f2,...), ;
```

b-str, e-str	Character string, to appear in the file, indicating the start and end of data of each particle. At least one of these must be specified.
t-str	Character string, to appear in the file, to terminate reading the file. Read to the end-of-file if not specified.
c	Character indicating the start of comment data (to the end of the file line)
kw_j	Name of variables which appear in the file. Case sensitive. Each data is expected in the form `a=3.0,` or `b=0.5,2.0,` etc. If a data has more than one components, you need to add the size. The above form requires the `KEYWORD` operand of the form `KEYWORD=(a, b(2)),`
x_i, f_i	Define the rule to convert the input data (`KEYWORD`s) into the **CAIN** variables. The l.h.s. x_i is either one of `T`,`X`,`Y`,`S`,`Px`,`Py`,`Ps`, `Sx`,`Sy`,`Ss`,`Xi1`,`Xi2`,`Xi3`, `Kind`,`Gen`,`Wgt`. The r.h.s f_i is an expression including the `KEYWORD`s kw_j's (and possibly other variables). You have to define all the 16 varables above. (Default is all zero except `Kind`=1, `Gen`=1.) Note that `En` does not appear here because the energy is computed from $\sqrt{m^2+\boldsymbol{p}^2}$.

An example. Suppose each data in the file, representing an electron at $s=0$, is given as

```
START  xy = 0.1, -0.5,  T=0.3, pxy=0.01, 0.03, dp=0.001  END
```

where `xy` is in mm, `T` in mm, `pxy` in milli-radian, `dp`= $\Delta p/p_0$ with p_0=1GeV/c. Then the `BEAM` command should be

```
SET   p0=1e9;
BEAM  FILE='...', USERDEFINED, BEGIN='START', END='END',
      KEYWORD=(xy(2), T, pxy(2), dp),
      CONVERSION=( X=xy(1)/1e3, Y=xy(2)/1e3, T=T/1e3, S=0,
      Ps=(1+dp)*p0, Px=pxy(1)*Ps, Py=pxy(2)*Ps, Kind=2, Wgt=1e6);
```

The spin will be zero. Note that the `CONVERSION` functions are evaluated in the given order so that, in this example, `Ps` is already defined when `Px` and `Py` are computed.

A complication arises when some of the above 16 variables appear as keywords. If they are used as exactly same meaning as in **CAIN** (including the unit system), you need (must) not list them as `KEYWORD` and must not give `CONVERSION` formula. If they are different, they have to appear in `KEYWORD` and in the l.h.s. of the `CONVERSION` formula. In this case the l.h.s. of a `CONVERSION` formula is a **CAIN** varable but the same name appears on the r.h.s. as a user keyword. In the above example, `T` in the file is in mm, different from **CAIN** definition so that it is listed in `KEYWORD` and the conversion formula `T=T/1e3` is given.

Tips

- Blanck spaces in the file data are treated as delimiters. Comma '`,`' and line feed are replaced by blanck spaces.
- A practical problem is the computing time since the file data has to be interpreted character by character (typically 0.5 to 1 msec per particle on 750MHz platform). If you are going to use the same particle data many times, you had better convert it into standard format. A **CAIN** job consisting of two commands
  ```
   BEAM FILE='file1.dat', USERDEFINED, ....;
   WRITE BEAM, FILE='file2.dat';
  ```
  will do.

3.5.3 Single particle

You can define (add to the existing list) one particle by explicitly specifying the data. It is not reccommended, however, to define many particles by this way because of the computing (interpreting) time.

Syntax:

`BEAM` `SINGLEPARTICLE,` `KIND=`k`,` `WEIGHT=`w`,` [`TXYS=(`t`,`x`,`y`,`s`),`]
`P=(`p_x`,`p_y`,`p_s`),` [`SPIN=(`s_x`,`s_y`,`s_s`),`] `;`

k	Particle specie.
w	Weight (how many real particles are represented).
t,x,y,s	Space-time location of the test particle (m). Default is (0,0,0,0).
p_x, p_y, p_s	3-momentum (eV/c). p_s must not be zero (i.e., either right-going or left-going). The energy is computed from $E^2 = m^2 + \boldsymbol{p}^2$.
s_x, s_y, s_s	3 components of the spin. Default is (0,0,0).

3.5.4 Test particles

Definition of test particles can also be done by `BEAM` command. One `BEAM` command is needed for each test particle. The number of test particles times the number of `PUSH` time steps must be less than 5000 (parameter `MTSTP` in the file 'cain/src/include/tstpcm.h'). Test particles do not create a field but feel a field. They do not interact with lasers and do not create particles (such as beamstrahlung, incoherent pair, etc). Therefore, 'test photon' does not make sense.

Coordinates and energy-momentum of test particles can be refered to at any time by functions `TestT`, etc. See Sec.2.6.

Syntax:

`BEAM` `TESTPARTICLE,` `NAME=`n`,`|'name'`,` `KIND=`k`,` [`TXYS=(`t`,`x`,`y`,`s`),`]
`P=(`p_x`,`p_y`,`p_s`),` [`SPIN=(`s_x`,`s_y`,`s_s`),`] `;`

n,name	A test particle must have a name, consisting of upto 3 characters. The 'name' (left-adjusted) must be enclosed by a pair of apostrophes. It can also be specified by an integer $-99 \leq n \leq 999$, which is converted to a decimal character string (right-adjusted). Thus, `NAME=1` and `NAME='  1'` is identical. (In the computer, one character '`T`' is added at the top. Thus, `NAME=999` becomes `T999`.)
k	Particle specie.
t,x,y,s	Location of the test particle (m). Default is (0,0,0).
p_x, p_y, p_s	3-momentum (eV/c). p_s must not be zero (i.e., either right-going or left-going).
s_x, s_y, s_s	3 components of the spin. Default is (0,0,0).

3.5.5 Caution

What is actually defined by the particle variables (t, x, y, s) and (E, p_x, p_y, p_s) is not a particle at a definite space-time coordinate, but rather is a straight trajectory (a world line) which passes the space-time point (t, x, y, s). At the time when the `PUSH` command is executed, they are first pulled to the intercept on $t = t_0$ plane, where t_0 is the starting time of the `PUSH` loop.

When a `BEAM` command is inserted within a `PUSH` loop, the particles are taken to the corresponding time t=`Time`. However, it is safer not to insert `BEAM` command within `PUSH` loop unless you know well what is going on. One exception is the test particles, which in some cases you want to create during a `PUSH` loop (for example to see the behavior of a low energy particle created during interaction). If you do not want them to be time-shifted in such cases, you can define the `TXYS` operand as `TXYS=(Time,...)`, where `Time` is the `PUSH` running time ('present time').

3.6 LASER

Defines a laser. There can be upto 5 lasers but this can easily be increased (parameter `MLSR` in `src/module/lasrdata.f`). One `LASER` command defines one laser. Note that lasers, if there are more than one, act incoherently. Their interference effects cannot be included in the present version.

The interaction with photons/electrons/pisitrons is defined by the command `LASERQED`.Sec.3.7

The parameters for a laser can be classified into 3 categories.

(a) General parameters such as wavelength, peak power density, and those defining laser coordinate sytem (τ, ξ, η, ζ). (See Sec.5.8.1 for laser coordinate). This category includes the parameters `RIGHT/LEFT`, `POWERDENSITY`, `WAVELENGTH`, `TXYS`, `E3`, `E1`, and `STOKES`.

(b) Parameters to specify time profile $F(\tau - \zeta)$ of the pulse. This includes `SIGT`, `TTOT`, `GCUTT`, and `TEDGE`.

(c) Parameters to specify spatial profile $F(\xi, \eta, \zeta)$ of the pulse. This includes `RAYLEIGH`, `GCUT`, and `TDL`, `DONUT` and those specifying the donut shape.

You can specify (b) and/or (c) by a file defined by the `FILE` operand. The laser profile is determined in the following order.

1. `FILE` operand has the highest priority. If time or space profile is missing, it is expected to be given by the command parameters.
2. The time profile, if not given in the file, can be Gaussian or trapezoidal according to whether `SIGT` or `TTOT` operand is specified.
3. The spatial profile, if not given in the file, is donut shape if `DONUT` operand is given, otherwise Gaussian.

3.6.1 General laser parameters

Syntax:

`LASER` [`RIGHT`|`LEFT`,] `WAVELENGTH=`λ_L`,` `POWERDENSITY=`P_{peak}`,`
[`TXYS=(`t`,`x`,`y`,`s`),`] `E3=(`$e_x^{(3)}$`,`$e_y^{(3)}$`,`$e_s^{(3)}$`),` `E1=(`$e_x^{(1)}$`,`$e_y^{(1)}$`,`$e_s^{(1)}$`),`
[`STOKES=(`ξ_1`,`ξ_2`,`ξ_3`),`] [`FILE=`f_n|'file_name'`,`] [`SHIFTT=`$\Delta\tau$`,`]
[`SHIFTZ=`$\Delta\zeta$`,`] [`PLOTPROFILE,`] [`TPROFILE=`τ_{prof}] |
[`TPROFILE=(`τ_{prof1}`,`τ_{prof2}`),NPROFILE=`n_{prof}`,`] [time_profile_params,]
[spatial_profile_params,] ;

`RIGHT`|`LEFT` Right-going or left-going. If `RIGHT`(`LEFT`) is specified, the laser acts only onto the left(right)-going particles (to save computing time). If omitted, acts on both.

λ_L Laser wavelength (m).

P_{peak} Peak power density (Watt/m^2). If `FILE` operand is used, P_{peak} is multiplied on the power density values obtained from the file.

t,x,y,s Laser focal point and the time when the laser pulse comes there (m).

$(e_x^{(3)}, e_y^{(3)}, e_s^{(3)})$ Unit vector $\boldsymbol{e}^{(3)}$ along the direction of laser propagation.

$(e_x^{(1)}, e_y^{(1)}, e_s^{(1)})$ Unit vector $\boldsymbol{e}^{(1)}$ perpendicular to $\boldsymbol{e}^{(3)}$. $(\boldsymbol{e}^{(1)}, \boldsymbol{e}^{(2)}, \boldsymbol{e}^{(3)})$ with $\boldsymbol{e}^{(2)} = \boldsymbol{e}^{(3)} \times \boldsymbol{e}^{(1)}$ forms a right-handed orthonormal frame. $\boldsymbol{e}^{(3)}$ and $\boldsymbol{e}^{(1)}$ need not be normalized exactly and need not be perpendicular to each other exactly (The component parallel to $\boldsymbol{e}^{(3)}$ is subtracted from $\boldsymbol{e}^{(1)}$ by Schmidt orthogonalization).

ξ_1,ξ_2,ξ_3 Stokes parameter defined in the $(\boldsymbol{e}^{(1)}, \boldsymbol{e}^{(2)}, \boldsymbol{e}^{(3)})$ frame. Default=(0,0,0).

f_n,file_name Unit number or name of an existing file. (For the differece between the two forms, See Sec.2.8.) See below for the file format.

$\Delta\tau$,$\Delta\zeta$ Add to the coordinate data (τ, ζ) of the file.

PLOTPROFILE Flag to plot the intensity profile on TDFile. This is basically to check the profile defined by a file. You cannot specify axes, scale, range, etc. The only parameter is τ_{prof}.
You can also plot the intensity profile as a scatter plot of laser photons by using PLOT SCAT, LASERPHOTON command. (See Sec.3.24.2)

τ_{prof} The time (multiplied by velocity of light) to take snapshop for plotting laser profile. In units of meter in the laser coordinate (τ, ξ, η, ζ).

τ_{prof1},τ_{prof2},n_{prof} Number and time range for plotting profile.

See Sec.5.8.1 for more detail about the laser definition.

3.6.2 Time profile parameters

Syntax:

SIGT=σ_τ|TTOT=τ_{tot}, [GCUTT=n_{tcut},] [TEDGE=τ_{edge},] ;

σ_τ R.m.s. pulse length (times velocity of light) in power, not in field amplitude, assuming Gaussian structure. (meter)

n_{tcut} Cut off of longitudinal tail in units of sigmas for Gaussian time structure. Default=3.5.

τ_{tot} Total pulse length for trapezoidal longitudinal structure (meter). Either one of SIGT or TTOT must be specified.

τ_{edge} Longitudinal edge length (meter) for trapezoidal time structure. The flat-top length is then $\tau_{tot} - 2\tau_{edge}$. Default=0 (i.e., rectangular shape).

3.6.3 Spatial profile parameters

Syntax:

[RAYLEIGH=(β_1,β_2),] [GCUT=n_{cut},] [TDL=(d_1,d_2),] [DONUT, AAXICON=a_{ax}, BAXICON=b_{ax}, FOCALLENGTH=f, SIGMA0=σ_0, RMAX=r_{max}, ZMAX=ζ_{max}] ;

β_1,β_2 Rayleigh length in $\boldsymbol{e}^{(1)}$, $\boldsymbol{e}^{(2)}$ direction. (meter)

n_{cut} Cut off of transverse tail in units of sigmas. Default=3.5.

d_1,d_2 Dilatation factor for a laser which is not at the diffraction limit. (TDL means 'times diffraction limit'.) This factor is multiplied to the emittance of the laser beam which is $\lambda_L/4\pi$ at the diffraction limit. See Sec.5.8.1 for more detail. Default is $d_1 = d_2 = 1$. Note that POWERDENSITY has to define the power density with the TDL parameters included.

`DONUT`	Flag for donut shape (created by an axicon). See Fig.5.5 for the definition of the geometric parameters. The operand `POWERDENSITY` in this case specifies the power density at the entrance of the axicon. The spatial profile at the entrance is assumed to be Gaussian defined by σ_0. The power (Watt) integrated over the transverse plane is $$P = 2\pi\sigma_0^2 P_{peak}\left(1 - e^{-(a_{ax}^2 - b_{ax}^2)/2\sigma_0^2}\right)$$
a_{ax},b_{ax}	Outer and inner radius of the axicon.
f	Focal length of the lens just after the axicon.
σ_0	Rms radius of the laser beam at the entrance of the axicon.
r_{max},ζ_{max}	Specify the range of laser field. $\sqrt{\xi^2+\eta^2} < r_{max}$ and $-\zeta_{max} < \zeta < \zeta_{max}$. A table will be created only in this region. (So, time consuming if unnecessarily large.)

3.6.4 File format

The file must be written in the following format.

- When a character "!" appears in a line, the rest of the line is considered as a comment.
- The first non-comment line must be

  `ORDER=a`

  where 'a' is a combination of characters `L`, `X`, `Y`, `Z`, `R`. These characters mean that the following table specifies the $\tau - \zeta$ (`L`), ξ (`X`), η (`Y`), ζ (`Z`), $\sqrt{\xi^2+\eta^2}$ (`R`) dependence of the power density. The possible combinations are

`L`	$F(\tau - \zeta)$
`XYZ`	$F(\xi, \eta, \zeta)$
`RZ`	$F(r, \zeta)$
`LXYZ`	$F(\tau - \zeta, \xi, \eta, \zeta)$
`LRZ`	$F(\tau - \zeta, r, \zeta)$

  or their permutation of characters. The number of characters is the dimension of the table. The order of the characters specify the nesting order in the table. For example, RZ implies the FORTRAN order `((F(IR,IZ),IR=1,M),IZ=1,N)`.
- The following lines define the range of the variables specified in `ORDER` in the form

  a= a_1 a_2 n_a

  where 'a' is one of the characters which appear in `ORDER`, $a_1(a_2)$ the first(last) value and n_a the number of values. (a_2 can be $< a_1$.) For example, if `ORDER=RZ`, the lines

  `R=` r_1 r_2 n_r

  `Z=` ζ_1 ζ_2 n_ζ

  are expected. (Input order of `R=` and `Z=` is irrelevant.)

- Optional lines to specify the units in the form

  a`FAC`=c

  where 'a' is one of the characters which appear in `ORDER` or '`P`' for power density (see below) and c is a number. These numbers are to be multiplied to get the coordinates in meter (or power density in Watt/m^2).
- Finally, power density data in the form

  `P` = $P_{111}\ P_{211}\ P_{311}\ \ldots$

  The data are separated by one or more blanck characters. There can be any number of carriage return unless a data is split into two lines.

  These must be aligned as defined by the character order in `ORDER`. The data are in units of Watt/m^2 (after multiplied by `PFAC` in the file and/or `POWERDENSITY` in the command). Linear interpolation is applied.
- For the case `ORDER`$\neq$`L`, you may optionally define the local propagation direction $\boldsymbol{n}$ in the form for `ORDER=XYZ` or `ORDER=LXYZ`

  `N=` $n_{\xi,111}\ n_{\eta,111}\ n_{\zeta,111}\ n_{\xi,211}\ n_{\eta,211}\ n_{\zeta,211}\ \cdots$

  or for `ORDER=RZ` or `ORDER=LRZ`

  `N=` $n_{r,111}\ n_{\zeta,111}\ n_{r,211}\ n_{\zeta,211}\ \cdots$

  If `N=` does not appear, $\boldsymbol{n} = (0, 0, 1)$ is used, which means parallel propagation along `E3` vector.
- Then, `ORDER` can appear again to define other dependences. For example, the first `ORDER` defines $F(\tau - \zeta)$ and the second defines $F(r, \zeta)$. Of course, there can be only one `ORDER` if it defines $F(\tau - \zeta, \xi, \eta, \zeta)$ or defines $F(\tau - \zeta, r, \zeta)$.

3.6.5 Laser-related CAIN functions

One can use the CAIN function `LaserIntensity` to retrieve the laser power density at a specified space-time point. (See Sec.2.6) This may be useful, e.g., when ξ_{max} is needed in `LASERQED` command.

Another CAIN function is `LaserRange` to retrieve the range of coodinate where the laser field is non-zero.

See Sec.2.6 for detail.

3.7 LASERQED

Defines the method and parameters for the calculation of the interaction between lasers and particles. See Sec.**??** sec:LinComptonSec.5.8.3 and Sec.5.8.4 for more detail on physics.

As of **CAIN**2.35 there can be only one `LASERQED` command for each of Compton and Breit-Wheeler processes. This means all the lasers must share the same `LASERQED` command, if there are more than one laser. Therefore, you cannot use circularly and linearly polarized lasers together if you want nonlinear interaction (`NPH`$\geq$ 1). This point will be improved in later versions when such a case becomes needed.

Syntax:

LASERQED COMPTON|BREITWHEELER[,]† [CIRCULARPOL|LINEARPOL,]
NPH=n_{ph}, [NY=n_y,] [NXI=n_ξ,] [NLAMBDA=n_λ,] [NQ=n_q,] XIMAX=ξ_{max},
LAMBDAMAX=λ_{max}, ETAMAX=η_{max}, [PMAX=p_{max},]
[ENHANCEFUNCTION=f_{enh},] [LENHANCE=l_{enh},] ;

COMPTON|BREITWHEELER Specifies which parameters to define here.

CIRCULARPOL|LINEARPOL Polarization of laser. Needed for $n_{ph} \geq 1$ only. Elliptic polarization is not ready when nonlinear effects are needed. **As of CAIN2.35 Breit-Wheeler process with linear polarization is not ready. Also note that the electron spin is ignored for linear polarization case.** Default is circular polarization.

n_{ph} Maximum number of laser photons to be absorbed in one process.
If < 0, turn off Compton or Breit-Wheeler.
If $=0$, use linear Compton/Breit-Wheeler formula.
If ≥ 1, use nonlinear formula.
Note that $n_{ph} = 0$ and $n_{ph} = 1$ are different. The former is the limit of $\xi \to 0$, which contains $n_{ph} = 1$ term only, whereas the latter is a truncation of the exact series with respect to n_{ph}. When $n_{ph} = 0$, none of the variables (n_y, n_ξ, n_λ, n_q, ξ_{max}, λ_{max}, η_{max}) are needed.
When $n_{ph} \geq 1$, only longitudinal polarization is considered and the lasers must be circularly polarized by 100% (i.e., $\xi_1 = \xi_3 = 0$, $\xi_2 = \pm 1$).

n_y Number of abscissa for final energy. Default=20.

n_ξ Number of abscissa for ξ parameter. Default=20.

n_λ Number of abscissa for λ parameter. Applies to Compton case only. Default=20.

n_q Number of abscissa for q parameter. Applies tp Breit-Wheeler case only. Default=50.

ξ_{max} Maximum value of ξ for the table.

λ_{max} Maximum value of λ for the table. Applies to Compton case only.

η_{max} Maximum value of η for the table. Applies to Breit-Wheeler case only.

p_{max} Maximum probability of events per one time step. If the computed probability exceeds p_{max}, **CAIN** of present version stops with a message.

f_{enh} Defines a function in order to artificially enhance the event rate. You can enhance a part of spectrum. It is defined as an expression containing Y as the final energy parameter ($0 \leq$Y≤ 1). Its value must be ≥ 1 for all Y. Generally speaking, Y close to 1 generates low energy charged particles. For example,

```
 ENH=1+Step(Y-0.8)*(Y-0.8)*10
```

will enhance the events with Y> 0.8 by a factor upto 3 (at Y=1).
In the program, the real spectrum function is multiplied by f_{enh} and, when an

event is generated, the created particles are asigned a weight $1/f_{enh}$.[1]
This function is used only during the initialization by LASERQED command. Therefore, if the expression contains user-defined parameters, their values at the time of LASERQED command are used. Changing them afterwards will not affect the computation.

l_{enh} Integer 1 or 2 or 3. Defines how to treat macro-particles under enhancement. Needed only when you specify f_{enh}. Normally you should use the default (=1) but you have to read below if you suffer from generating too many unnecessary particles. When an event is generated under $l_{enh} = 1$, **CAIN** generates new macro-particles with the weight w_0/f_{enh}, where w_0 is the weight of the initial particle, and leave the initial macro-particle with the reduced weight $w_0(1 - 1/f_{enh})$.
When f_{enh} is very large, however, in the case of Compton scattering, for example, this rule will create many photons and electrons with small weight. The electrons thus created cause another events generating electrons of even lower weights. If you are interested only in the photon, this is just a waste of computing time and memory. This problem can be avoided by l_{enh}=2 or 3.
In the case of Compton,

- If l_{enh}=2, the initial particle ($e^{\pm}$) is treated probabilistically, i.e., it is replaced by the new $e^{\pm}$ with the probability $1/f_{enh}$, but remain as it was with the probability $1 - 1/f_{enh}$. Therefore, the number of macro-$e^{\pm}$ will never increase.
- If l_{enh}=3, the new particle (γ) is treated probabilistically, i.e., a γ of weight w_0 (not w_0/f_{enh}) is generated with the probability $1/f_{enh}$. Thus, the number of macro-γ is the same as in the absense of enhancement.
- To treat both initial and new particles probabilistically is meaningless (doesn't enhance the events).

If you are interested only in the photons ($e^{\pm}$), l_{enh}=2 (3) is recommended.
In the case of Breit-Wheeler,

- If l_{enh}=2, the initial particle (γ) is treated probabilistically, i.e, it is eliminated with the probability $1/f_{enh}$, but remain as it was with the probability $1 - 1/f_{enh}$.
- If l_{enh}=3, the new particles ($e^{\pm}$) are treated probabilistically, i.e, a pair of the weight w_0 is generated with the probability $1/f_{enh}$.

3.8 CFQED

Constant-Field QED, i.e., the beamstrahlung and coherent pair creation. Both the effects of the beam field and the external field are included. The angular distribution of the final

[1] Note that f_{enh} slightly larger than 1 is useless (even harmful) because a small fraction $1 - 1/f_{enh}$ of the parent particle will remain as a macro-particle, causing a waste of computing time. In the example above, $f_{enh} = 1$ exactly for Y<0.8.

particles is not included.

When the polarization flag (see below) is on, all the polarization effects (longitudinal and transverse spin of electron/positron and linear and circular polarization of photon) are included.

Syntax:

CFQED BEAMSTRAHLUNG|PAIRCREATION[,]† [POLARIZATION,]
  [PMAX=p_{max},] [WENHANCE=w_{enh},] ;

BEAMSTRAHLUNG|PAIRCREATION Specifies which parameters to define here. Only one of these may be specified by one CFQED command.

POLARIZATION Flag to take into account all the polarization effect. (default=No). Note that the flag SPIN (FLAG command) must also be on for polarization calculation.

p_{max} Maximum probability of events per one time step. (Default=0.1). When the probability exceeds p_{max}, **CAIN** stops with a message.
(This has been improved for PAIRCREATION after CAIN2.1e by dividing the time step into substeps so that the probability in a substep does not exceed p_{max}.)

w_{enh} Enhancement factor of radiation rate. $0 \leq w_{enh}$. When $w_{enh} = 1$ (default), macro-photons are created such that $n_{\mathrm{macro}\gamma}/n_{\mathrm{macroe}} = n_{\mathrm{real}\gamma}/n_{\mathrm{reale}}$
When $w_{enh} > 1$ (< 1), macro-photons are created more (less) by the factor w_{enh}, each having less (more) weight. When $w_{enh} = 0$, no photon is created (but the recoil of electron is taken into account.) This operand is introduced in order to avoid poor statistics due to too less macro-photons or memory overflow due to too many macro-photons.

See Sec.5.9 and Sec.5.10 for the formulas and algorithm and for more detail on the enhancement factor.

3.9 BBFIELD

Define the parameters for the calculation of beam-beam field.

Syntax:

BBFIELD WX=w_{x1}|WX=(w_{x1}[,w_{x2}]), [WXMAX=w_{xm1}|WXMAX=(w_{xm1},w_{xm2}),]
  R=r, [NX=n_x,] [NY=n_y,] [PSIZE=Δ,] [NMOM=n_{mom},] ;

w_{x1},w_{x2} Horizontal width of the mesh in meters for right and left-going beams. If w_{x2} is not specified, $w_{x2} = w_{x1}$ is adopted. No default for w_{x1}.

w_{xm1},w_{xm2} If WXMAX is given, the with of the mesh region can vary in the range (w_x, w_{xm}) when the beam fraction outside the range defined by WX and R is significant. Note $w_{xm} \geq w_x$.

r	Aspect ratio $(w_x/n_x)/(w_y/n_y)$ of the horizontal to vertical mesh size. This is common to right and left-going beams. No default.
n_x,n_y	Number of horizontal and vertical bins. Present version uses Fast Fourier Transformation so that a power of 2 is the best choice. Other numbers are also allowed but those of the form 2^n or 3×2^n or 5×2^n are recommended. Default=32.
Δ	Macro-particle size in units of the bin size. Macro-particles are treated as a rectangular of uniform distribution. Must be $0 \le \Delta \le 1$. Default=1.
n_{mom}	For (x, y) points outside the mesh region, a harmonic expansion using the elliptic coordinate is used. The parameter n_{mom} specifies the truncation of harmonics. $n_{mom} = 0$ takes only the total charge term and $n_{mom} < 0$ ignores the field outside. Default=10. Note that the particles outside mesh region receive the beam-beam kick unless $n_{mom} < 0$, but the field created by them is not taken into account. See Sec.5.7 for more detail.

Note that the longitudinal mesh size, which is common to beam-beam field and luminosity calculations, has to be defined by the parameter `Smesh` by the `SET` command. Its value at the time when `PUSH` started is used thoughout the `PUSH` loop.

3.10 EXTERNALFIELD

Define external field. The present version allows only a constant field over an interval bordered by two parallel planes.
Syntax:

`EXTERNALFIELD` [`S=(`s_1`,`s_2`),`] [`V=(`c_x`,`c_y`,`c_s`),`] [`E=(`E_x`,`E_y`,`E_s`),`]
[`B=(`B_x`,`B_y`,`B_s`),`] `;`

s_j, c_j	Define the range of the field as $s_1 \le c_x x + c_y y + c_s s \le s_2$. Must be $s_1 < s_2$. Default $s_1 = -\infty$, $s_2 = +\infty$ and (c_x,c_y,c_s)=(0,0,1).
E_j	Electric field components in units of V/m. Default=(0,0,0).
B_j	Magnetic field components in units of Tesla. Default=(0,0,0).

3.11 LUMINOSITY

Define the transverse mesh size, number of bins, etc, for luminosity calculation. One luminosity command is needed for each combination of particles γ, e$^-$, e$^+$, right-going and left-going. Thus, there can be at most 9 `LUMINOSITY` commands.

Syntax:

```
LUMINOSITY    KIND=(k_r,k_l),    [FREP=f_rep,]
   [W=(W_min,W_max,n_bin),|W=(W_0,W_1,...,W_n_bin),|W=w_array,]
   [E1=(E_1min,E_1max[,n_1bin]),|E1=(E_1,0,E_1,1,...,E_1,n_1bin),|E1=e_1array,]
   [E2=(E_2min,E_2max[,n_2bin]),|E2=(E_2,0,E_2,1,...,E_2,n_2bin),|E2=e_2array,]
   WX=(w_x[,w_xm]),   WY=(w_y[,w_ym]),   [HELICITY,]   [ALLPOL,] ;
```

k_r,k_l Particle species of right and left-going beams.

f_{rep} Repetition frequency (Hz). Used for the luminosity scale only. Default=1Hz.

W_{min},W_{max},n_{bin} Parameters for differential luminosity with respect to the center-of-mass energy W. (W_{min},W_{max}) is the range in eV and n_{bin} is the number of bins. If (W_{min},W_{max}) is not given, the center-of-mass spectrum is not calculated. Default for n_{bin} is 50.

$W_0 \ldots W_{n_{bin}}$ Define the center-of-mass energy bins in the case of non-equal spaced bins. n_{bin} is the number of bins, W_0 is the lower edge of the first bin and $W_{n_{bin}}$ is the upper edge of the last bin. n_{bin} must be ≥ 3 in order to distinguish from the equal-space case. n_{bin} must be ≤ 200.

w_{array} The above two cases can also be specified by a name of an array w_{array}. It must be one dimensional array with no subscript. Its full size is used. If the size is $\leq$ 3, it is understood as (W_{min},W_{max},n_{bin}), otherwise as ($W_0 \ldots W_{n_{bin}}$).

E_{1min},E_{1max},n_{1bin},E_{2min},E_{2max},n_{2bin} 2-D differential luminosity $d\mathcal{L}/dE_1dE_2$. (E_{jmin}, E_{jmax}) is the range in eV and n_{jbin} is the number of bins. (j=1 for right-going beam and j=2 for left-going beam.) Both or none of `E1` and `E2` have to be specified. If none is specified, 2-D luminosity is not calculated. Default for ni, bin is 50.

$E_{i,j}$,$n_{i,bin}$ $(i = 1, 2)$ Define non-equal spaced bins. Similar to the case of the center-of-mass energy.

e_{1array},e_{2array} One-dimensional array names without subscripts. Similar to w_{array}.

w_x,w_y Full horizontal/vertical width of the mesh region (m). The origin is adjusted automatically from time to time.

w_{xm},w_{ym} Maximum width of the mesh region (m). If not given, $w_x(w_y)$ is used throughout. If given, an increased size upto $w_{xm}(w_{ym})$ is used when a significant particle fraction gets out of the mesh region defined by (w_x, w_y). The number of mesh points is determined automatically.

`HELICITY` Calculate luminosity for every combination of helicity, (++), (−+), (+−), (−−).

`ALLPOL` Calculate luminosity for all possible 16 combinations of the spins. (see Sec.5.6.2 for detail.)

All the LUMINOSITY commands must have the same value of w_x,w_y,w_{xm},w_{ym}, and f_{rep}. (Specify them at the first LUMINOSITY command.)

Note that the longitudinal mesh size, which is common to beam-beam field and luminosity calculations, has to be defined by the parameter Smesh by the SET command.

The luminosity is actually computed by the PUSH-ENDPUSH loop. The calculated luminosity can be referred to by the following functions. (If during the loop, the accumulated luminosity upto that moment is returned.)

Lum(k_r,k_l)	Luminosity of KIND=(k_r,k_l) in units of cm^{-2}sec^{-1}.
LumH(k_r,k_l,h)	Helicity luminosity: helicity combination $(++)$ $(h=1)$, $(-+)$ $(h=2)$, $(+-)$ $(h=3)$, $(--)$ $(h=4)$. $h=0$ will give the total luminosity Lum(k_r,k_l). (cm^{-2}sec^{-1})
LumP(k_r,k_l,s_1,s_2)	Polarization luminosity. $(0 \le s_1 \le 3,\ 0 \le s_2 \le 3)$ See Sec.5.6.2 for definition. (cm^{-2}sec^{-1})
LumW(k_r,k_l,n)	Differential luminosity in the n-th bin. (cm^{-2}sec^{-1}/bin)
LumWbin(k_r,k_l,n)	Bin center (eV) of the n-th bin. If $n=0$, the number of bins is returned. (Error if $n<0$ or n is larger than the number of bins.)
LumWbinEdge(k_r,k_l,n)	Bin edge (eV) of the n-th bin. $(0 \le n \le$ number of bins. $n=0$ is the lower edge of the first bin and $n=$number of bis is the upper edge of the highest bin. (Error if $n<0$ or n is larger than the number of bins.)
LumWH(k_r,k_l,n,h)	Differential helicity luminosity. (cm^{-2}sec^{-1}/bin)
LumWP(k_r,k_l,n,s_1,s_2)	Differential polarization luminosity. $(0 \le s_1 \le 3,\ 0 \le s_2 \le 3)$ See Sec.5.6.2 for definition. (cm^{-2}sec^{-1}/bin)
LumEE(k_r,k_l,n_1,n_2)	2-D differential luminosity $d\mathcal{L}/dE_1dE_2$ for the bin (n_1,n_2). (cm^{-2}sec^{-1}/bin)
LumEEbin(k_r,k_l,l,n)	Bin center (eV) of the n-th bin of E_1 (l=1) or E_2 (l=2). If $n=0$, the number of bins is returned. (Error if $n<0$ or n is larger than the number of bins.)
LumEEbinEdge(k_r,k_l,l,n)	Bin edge of the n-th bin of E_1 (l=1) or E_2 (l=2). See LumWbinEdge for the definition of n.
LumEEH(k_r,k_l,n_1,n_2,h)	2-D differential helicity luminosity
LumEEP(k_r,k_l,n_1,n_2,s_1,s_2)	2-D differential polarization luminosity.

These functions can be included in expressions. Thus, you can write the computed luminosity on a file. In particular, the only way to retrieve the 2-D luminosity $d\mathcal{L}/dE_1dE_2$ is to use the above functions because PLOT LUMINOSITY command cannot plot it (KEK TopDrawer cannot draw 3-D plot). So, for example, to write e$^+$e$^-$ luminosity, say

```
 SET m1=LumEEbin(2,3,1,0), m2=LumEEbin(2,3,2,0);
 WRITE ((LumEE(2,3,n1,n2),n1=1,m1),n2=1,m2),
   FORMAT=(...);
```

If you are satisfied with a pre-defined format, you can use PRINT/WRITE LUMINOSITY command.

3.12 PPINT

Incoherent particle-particle interaction such as incoherent pair creation and bremsstrahlung.[2] The following processes are included:

Breit-Wheeler	$\gamma + \gamma \rightarrow \mathrm{e}^- + \mathrm{e}^+$
Bethe-Heitler	$\gamma + \mathrm{e}^\pm \rightarrow \mathrm{e}^\pm + \mathrm{e}^- + \mathrm{e}^+$
Landau-Lifshitz	$\mathrm{e} + \mathrm{e} \rightarrow \mathrm{e} + \mathrm{e} + \mathrm{e}^- + \mathrm{e}^+$
Bremsstrahlung	$e + e \rightarrow e + e + \gamma$

All the processes except for Breit-Wheeler are calculated using the virtual photon approximation.

The circular polarization effect of the initial photons is included in the Breit-Wheeler process but all other polarization effects are ignored.

Particles created by incoherent processes do not contribute in creating the beam field. Also note that the parent macro-particles do not change by particle-particle interaction. All these come from the actual situation in linear colliders where the incoherent particles are much less in number compared with the initial particles.

Syntax: Specify virtual photon options

PPINT VIRTUALPHOTON[,]† [LOCAL,] [FIELDSUPPRESSION,] [EMIN=E_{min},] ;

LOCAL	Flag to adopt local virtual photon, i.e., to ignore the effects due to the finite transverse extent of virtual photons. Default is non-local.
FIELDSUPPRESSION	Flag to include the virtual-photon suppression effect due to strong external fields (normally, the beam-field by the on-coming beam). This can be effective when LOCAL is not specified. See section 3.4 of [6]. Default: does not include this effect.

E_{min}	Minimum energy of final electron/positron energies in eV. Default is twice the rest mass of electron. = 1.022..E6. This parameter is not directly related to virtual photons but included here because it is common to all the processes. The purpose of this parameter is to save computing time. The creation of pairs does not take too much computing time but to track extremely low-energy pairs in a strong beam field is very expensive. The worst ones are the pair particles having the sign of charge opposite to that of the on-coming beam because they are trapped in the strong field region. If you are not interested in them, you can eliminate them during the PUSH loop as
	CLEAR BEAM, INCP, RIGHT, KIND=2;
	CLEAR BEAM, INCP, LEFT, KIND=3;
	if the right(left)-going beam is electron(positron).

Syntax: Specify individual processes

PPINT BW|BH|LL|BREMSSTRAHLUNG[,]† [RIGHT,] [LEFT,]
[ENHANCE=f_{enh},] ;

[2] There is a known bug: PPINT is not effective when LUMINOSITY command is not invoked.

`BW,BH,LL,BREMSSTRAHLUNG` Specify one of Breit-Wheeer, Bethe-Heitler, Landau-Lifshitz, and Bremsstrahlung interactions. If more than one of these are needed, apply `PPINT` command repeatedly. No default.

`RIGHT,LEFT` Applies to Bethe-Heitler and Bremsstrahlung. The Bethe-Heitler process has two possible combinations, namely, (γ,e$^{\pm}$) and (e$^{\pm}$,γ). `RIGHT/LEFT` option specifies the photon is right-going or left-going or both. Default is both.
The Bremsstrahlung is treated as the interaction between real e$^{\pm}$ and a virtual photon. Therefore, it also has two possible combinations. This operand specifies which beam is the real particle.

f_{enh} Event rate enhancement factor. It is unity when the number of created macro-pairs is the same as the expected number of real pairs, *i.e.*, the weight of the pair particle is $1/f_{enh}$. Default f_{enh}=0.1.

In using **ABEL** one had to define the minimum scattering angle and minimum transverse momentum. This was due to the ultra-relativistic approximation employed there. **CAIN** does not need these parameters.

3.13 PUSH, ENDPUSH

Define the time step loop of tracking. Tracking is done by a pair of commands instead of one single command in order to allow users to take action such as print, plot, insert test particles, etc, at arbitrary time steps.
Syntax:

```
PUSH      Time=(t0,tf,nt) ;
... any commands ...
ENDPUSH      ;
```

t_0,t_f Start and end time (multiplied by velocity of light) of tracking (meter). Note the spelling of `Time` which contains lower case alphabet in contrast to other operand keywords consisting of upper case letters only. In fact, `Time` is a pre-defined variable name. Therefore, you can, for example, print its current value during `PUSH` loop by `PRINT Time, FORMAT=`....

n_t Number of time steps. (≥ 0)

Actual control of the loop is done in the following way.

- Before the first time step, all the particles are made to drift to $t = t_0$ (by straight lines).
- At the `PUSH` command of j-th loop ($j = 0, 1, \ldots, n_t$), the time variable `Time` is set to $t_j = t_0 + j\Delta t$ where $\Delta t = (t_f - t_0)/n_t$.
- Execute commands between `PUSH` and `ENDPUSH`.

- Control comes to `ENDPUSH`. If $j < n_t$, make tracking (beam-beam, beamstrahlung, laser interaction etc) for the time step $t_j \leq$ `Time` $\leq t_{j+1}$.
- If $j < n_t$, returns to `PUSH`.

Note that the commands between `PUSH` and `ENDPUSH` are executed $n_t + 1$ times. If $n_t = 0$, the actions taken are to drift all particles to t_0 and to do commands between `PUSH` and `ENDPUSH` once. If $n_t < 0$, **CAIN** stops at `PUSH` with an error message rather than at `ENDPUSH`.

3.14 DRIFT

Drift the particles to a certain time or to a certain s coordinate.
Syntax:

```
DRIFT    T=t1|DT=Δt|S=s1,   [RIGHT,]   [LEFT,]   [KIND=k|(k1,k2),]
  [EXTERNALFIELD,] ;
```

t_1	Drift until `Time`=t_1 (meter).
Δt	Drift over time interval Δt (meter).
s_1	Drift to s coordinate = s_1 (meter). In any of the three cases `T`, `DT`, and `S`, the particles may go backwards in time depending on the parameters.
`RIGHT`,`LEFT`	Drift right- or left-going particles only.
k	Drift only particles of kind k.
`EXTERNALFILED`	Take into account the external field.

When there is only external field without beam interaction, `DRIFT EXTERNAL` is much better (more accurate and faster) than the `PUSH` command. The difference is that `DRIFT EXTERNAL` uses an exact solution in a constant field whereas `PUSH` carries out step-by-step integration, and that `PUSH` accepts only t as the independent variable while `DRIFT` also allows s (as in most accelerator program codes).

How to use `DRIFT EXTERNAL` may be understood by the following example. Suppose that the region $s_1 < s < s_2$ is shined by a laser. An electron beam comes from the left and goes through the laser region to created back-scattered photons, and subsequently goes through a magnetic field region $s_3 < s < s_4$. If the interval (s_2, s_3) is shorter than the bunch length, the bunch head is already in the field region when the tail gets out of the laser region. If you use `PUSH` command, you have to track the beam till the end of the magnetic field region. Instead, you can do more elegantly,

`BEAM .....`	Define electron beam
`LASER .....`	Define laser
`LASERQED .....`	Define laser QED parameters
`PUSH ....`	Start push without magnetic field
`ENDPUSH;`	End push
`EXTERNALFIELD ....`	Define external field
`DRIFT S=`s_3`;`	Pull back the beam to the plane s_3
`DRIFT S=`s_4`, EXTERNALFIELD;`	Calculate the field effects

3.15 LORENTZ

Coordinate transformation (shift of origin, rotation, Lorentz transformation) of particle coordinate, energy-momentum, polarization, etc. Using this command, you can transform a collision at an angle into a head-on collision.
Syntax:

`LORENTZ` [`TXYS=(`Δt`,`Δx`,`Δy`,`Δs`),`] [`ANGLE=`ϕ`,`] [`BETAGAMMA=`β_γ`,`]
[`AXIS=(`a_x`,`a_y`,`a_s`),`] [`EV=(`e_{vx}`,`e_{vy}`,`e_{vs}`),`] [`NOBEAM,`] [`RIGHT,`] [`LEFT,`]
[`KIND=`k|`(`k_1`,`k_2`),`] [`EXTERNALFIELD,`] [`LASER,`] ;

$(\Delta t, \Delta x, \Delta y, \Delta s)$	Shift of origin. (m)
ϕ	Spacial rotation angle (radian). (rotation of the coordinate axis.)
β_γ	Lorentz boost parameter $\beta \times \gamma$. (Boost of the coordinate axis).
(a_x, a_y, a_s)	Unit vector along the rotation axis. Need not be normalized exactly.
(e_{vx}, e_{vy}, e_{vs})	Unit vector along the boost direction. Need not be normalized exactly.
`NOBEAM`	No transformation of particles. If specified, `RIGHT`, `LEFT`, `KIND` operands are ignored.
`RIGHT,LEFT`	Select right- or left-going particles only. If omitted, both are transformed.
k	Select only particles of kind k. If omitted, all species are transformed.
`EXTERNALFIELD`	Lorentz transformation of external field (transformation of the field strength and the boundary).
`LASER`	Lorentz transformation of lasers.

The three types of transformations are carried out in the order of the input keywords `TXYS`, `ANGLE`, `BETAGAMMA`. With one `LORENTZ` command, each transformation can be specified at most once.

Note that, for any type, the transformation is that of the coordinate axis rather than the particles themselves. Thus, for example, if you say `TXYS=(0,0,0,1)`, then the s-coordinate of the particles decreases by 1 meter.

3.16 MAGNET

Defines a magnet to be used in `BEAMLINE` command. Here 'MAGNET' means any element in beamlines such as magnets, drfit spaces, rf cavities, markers, etc., although at present limited to those expressed by the following parameters. One `MAGNET` command is needed for each element.

Syntax:

`MAGNET` 'name', [`LENGTH=`l`,`] [`ANGLE=`θ`,`] [`EDGE=(`ϵ_1`,`ϵ_2`),`] [`K1=`k_1`,`] [`ROTATE=`ϕ`,`] [`APERTURE=(`a_x`,`a_y`),`] [`RECTANGULAR,`] ;

name	Magnet name. Must be enclosed by apostrophes.[3] (General form of character expression not accepted.) Up to 8 characters containing only upper and lower alphabet, numbers (0 to 9) and underscore '_'. Redefined if already exists. (You cannot eliminate the magnet name once defined.)
l	Magnet length in meter. Default=0. Can be a meta-expression. See below.
θ	Horizontal orbit bending angle in radian. (positive for bend to $-x$ direction.) Default=0. l cannot be zero if $\theta \neq 0$ (thin lens bend not allowed).
ϵ_1,ϵ_2	Entrance and exit edge angle in radian. Zero for sector-type magnets and $\epsilon_1 = \epsilon_2 = \theta/2$ for rectangular bend. Hard edge assumed (equivalent to a thin lens quad with $k_1 = -\theta \tan \epsilon / l$). The edge angle is effective only for thick lens bend (i.e., $l > 0$ and $\theta \neq 0$). Default=(0,0).
`RECTANGULAR`	Rectangular bend, i.e., $\epsilon_1 = \epsilon_2 = \theta/2$. Effective when $\theta \neq 0$. `EDGE` need not be specified.
k_1	Focusing strength defined by $l \times (e/p)\partial B_y/\partial x$ where e and p are charge and mementum of reference particle. This is equal to the inverse focal length. Positive for horizontally focusing magnets. Default=0. Thin lens is expressed by l=0. Can be a meta-expression. See below.
ϕ	Rotation of the magnet around the orbit axis in radian. Vertical bends (skew quads) are expressed by `ANGLE` (`K1`) and `ROTATE`. A bend with $\phi = +\pi/2$ bends the orbit downward ($-y$ direction). Default=0.
(a_x, a_y)	Magnet half aperture (m). Not checked if 0. Default=(0,0).

At present the following cannot be included.

- Combined function bends.
- Skew quads can be used in tracking by `TRANSPORT` command but the `BLOPTICS` cannot handle them.
- Errors of magnets.

[3]You may omit apostrophes in most cases but an error is caused if the name conflicts with the operand names (e.g., `L` is understood as the abbreviation of `LENGTH`).

- Higher multipoles.
- RF components.

An element with all zero parameters is possible. It may be used as a marker.

The maximum number of magnets is defined by the parameter MMAG in the ALLOCATE command. The default is 200.

Meta-expression

The length l and gradient k_1 can be a meta-expression (see Sec.2.7). It is re-evaluated when the parameter changes. For example,

```
  SET k=0.1;
  MAGNET 'QFH', K1=k/2;
```

will set the parameter K1 to be 0.05 but in the case

```
  SET k=0.1;
  MAGNET 'QFH', K1='k/2';
```

the expression k/2 itself is stored with its initial value 0.05 and is re-evaluated when k changes (for example during matching). You can also write, if you want, as

```
  SET $qfh='k/2', k=0.1;
  MAGNET 'QFH', K1=$qfh;
```

3.17 BEAMLINE

Defines a beamline consisting of magnets. To use a beamline, see TRANSPORT command (Sec.3.20). To calculate the optics see BLOPTICS command (Sec.3.18).

Syntax:

BEAMLINE 'name', LINE=(name$_1$, name$_2$, ..., name$_n$) [APERTURE=(a_x,a_y),] ;

name	Beamline name. Same rule as the magnet naming rule. Must be enclosed by apostrophes. (General form of character expression not accepted.) Re-defined if already exists. (You cannot eliminate the beamline name once defined.) **Present version does not work correctly if re-defined. Sorry.**
name$_j$	Name of already defined magnet or beamline. You can put − sign for a reversed beamline. You cannot use nested ( ). If needed, you must asign a name for the part of the beamline.
(a_x, a_y)	Half aperture for the whole beamline (m). Not checked if 0. Default=(0,0). This value is overwritten if its components (beamline or magnet) have their own aperture.

BEAMLINE command only defines the sequence of magnets. It does not define the location in the CAIN coordinate and does not have information of the beam. They must be defined when used in TRANSPORT command.

You can print/plot the optics by `PRINT/PLOT BLOPTICS` command. But prior to these, you need to call `BLOPTICS` command. Also, you can print the geometry of a beam line by `PRINT BLGEOMETRY` command.

The maximum number of beamlines is defined by the parameter `MBEAMLINE` in the `ALLOCATE` command. The default is 50.

3.18 BLOPTICS

Calculate Twiss parameters of a beamline. This command is needed only when you print/plot optics by using `PRINT BLOPTICS` or `PLOT BLOPTICS`. It is not used in particle tracking.

You need to specify either `PERIODIC` or the Twiss parameters at the entrance of the beamline.

Syntax:

`BLOPTICS` `BEAMLINE='name',` [`PERIODIC,`] [`BETA=(`β_x`,`β_y`),`] [`ALPHA=(`α_x`,`α_y`),`] [`ETA=(`η_x`,`η_y`),`] [`ETAPRIME=(`η'_x`,`η'_y`),`] `;`

name	Name of a defined beamline. The apostrophes can be omitted.
`PERIODIC`	Periodic condition (i.e., Twiss parameters at the entrance equal to those at exit) is applied. (`BETA` etc are not needed)
β_x,β_y,α_x,α_y,η_x,η_y,η'_x,η'_y	Twiss parameters and eta functions at the entrance. The default values are all zero so that at least β_x and β_y must be defined if not `PERIODIC`.

3.19 MATCHING

Optics matching of a beamline.

Syntax:

`MATCHING` `BEAMLINE='name',` [`PERIODIC,`] [`BETA=(`β_x`,`β_y`),`] [`ALPHA=(`α_x`,`α_y`),`] [`ETA=(`η_x`,`η_y`),`] [`ETAPRIME=(`η'_x`,`η'_y`),`] `VARIABLE=(`v_1`,`v_2`,...),` [`ZERO=(`f_1`,`f_2`,...),`] [`POSITIVE=(`g_1`,`g_2`,...),`] [`MINFUN=`h`,`] `;`

name	Name of a defined beamline.
`PERIODIC`	Periodic condition (i.e., Twiss parameters at the entrance equal to those at exit) is applied. (`BETA` etc are not needed)
β_x,β_y,α_x,α_y,η_x,η_y,η'_x,η'_y	Twiss parameters and eta functions at the entrance. The default values are all zero so that at least β_x and β_y must be defined if not `PERIODIC`. These expressions can be floating or character. If floating, the value is computed when `MATCHING` command is invoked and is treated as a

constant during matching. If character, the string is treated as a meta-expression (see Sec.2.7) and is repeadtedly evaluated during matching. Therefore, if the expression contains a variable which is to be treated as a matching variable, you have to enclose the whole expression to make it a meta-expression.

v_i Name of a variable (scalar floating variable or an element of a floating array or an array name without subscripts) to be treated as a matching variable. If it is an array name, all the elements of the array are treated as independent variables. The variables must be defined prior to `MATCHING` command. The values are used as the initial value for fitting.

f_i Expressions to be made zero. Floating or character. Always treated as a meta-expression. The number of `ZERO` conditions must not exceed the number of variables.

g_i Expressions that must be ≥ 0. Floating or character. Always treated as a meta-expression.

h Function to be minimized. Actually, **CAIN** tries to minimize $h + c^2$, where c is defined below.

The convergence is defined by

$$c = \left\{ \sum f_i^2 + \sum [\theta(-g_i) g_i]^2 \right\}^{1/2} \qquad (\theta\text{: step function})$$

CAIN stops if an error occurs during reading the input command. When the fitting somehow finishes, the above value is set to the predefined parameter `Convergence`. You have to decide to continue or to `STOP;` by checking this variable.

Also note that the beamline optics is calculated in the case when `Convergence` is set. You need not call `BLOPTICS` command.

Example

Following example is a FODO cell of phase advance 90 degrees with the quadrupole strengths as the matching variables. Starting from the thin lens approximation, try matching and print the variables (before and after matching) and then print the optics.

```
SET nu=0.25, lq=0.5, ldrift=4.5, lcell=2*(lq+ldrift),
    kf=4/lcell*Sin(Pi*nu), kd=-kf;
PRINT kf, kd;
MAGNET 'L', L=ldrift;
MAGNET 'QFH', L=lq, K1='kf/2';
MAGNET 'QDH', L=lq, K1='kd/2';
BEAMLINE  'HALFCELL', LINE=(QFH, L, QDH);
BEAMLINE  'CELL', LINE=(HALFCELL, -HALFCELL);
MATCHING  BEAMLINE='CELL', PERIODIC, VARIABLE=(kf,kd),
          ZERO=(Nu(1,99)-nu, Nu(2,99)-nu);
PRINT kf, kd;
IF Convergence>1e-4; STOP; ENDIF;
PRINT BLOPTICS, BEAMLINE='CELL';
```

(The second argument of `Nu` should be larger than the number of magnets so that it is understood as the beamline end.)

Caution and Tips

- Don't forget to enclose the magnet parameters by apostrophes if they contain matching variables. (See meta-expression in Sec.3.16) As of **CAIN**2.3 only the length `L` and focal length `K1` of magnets can contain matching variables. When you allow a length of a magnet to vary, you should impose the positive definite constraint by `POSITIVE` operand.
- The minimization procedure uses derivatives (by difference) of the constraint functions, assuming they are smooth. So, for example, if you want to limit the maximum strength of a quad, you should not write
  ```
  SET k=0.1, kmax=0.2;
  MAGNET 'QF', K1='k';
  .......
  MATCHING ......, POSITIVE=(kmax-Abs(k));
  ```
  but should write
  ```
  MATCHING ......, POSITIVE=(kmax^2-k^2);
  ```
  or
  ```
  MATCHING ......, POSITIVE=(kmax-k, kmax+k);
  ```
- There is a problem in the constraint on phase advance (common to all matching codes), which can rigorously be defined only up to modulo 2π. If you want to set the phase advance $/2\pi$ at `QF` to be 0.2 for example, you can write `ZERO=(Nu(1,'QF')-0.2)`. If this does not work, try either
  ```
  ZERO=(Cos(2*Pi*Nu(1,'QF'))-Cos(2*Pi*0.2)
  ```
  or
  ```
  ZERO=(Sin(2*Pi*Nu(1,'QF'))-Sin(2*Pi*0.2)
  ```
  depending on the region of the phase.
- **CAIN** tries to minimize $\sum f_i^2 + \sum[\theta(-g_i)g_i]^2$ (θ: step function). The expressions in `ZERO` and `POSITIVE` should properly be weighted.
- `MATCHING` stops when the beamline is unstable during matching in the case of `PERIODIC`. This problem may be solved by the following way, though much more tedius,
  ```
  SET bx=1, by=1, ax=0, ay=0;
  MATCHING BEAMLINE=..., VARIABLE=(bx,by,ax,ay),
  BETA=('bx','by'), ALPHA=('ax','ay'),
  ZERO=(Beta(1,99)-bx, Beta(2,99)-by, Alpha(1,99)-ax, Alpha(2,99)-ay);
  ```
  (η and η' are omitted in this example.) Don't forget to enclose expression in `BETA` and `ALPHA`.

3.20 TRANSPORT, ENDTRANSPORT

`PUSH-ENDPUSH` loop can track the particles under various interactions but it is not convenient for tracking the individual particle motion in magnetic beamlines. (`DRIFT EXTERNAL` command can do this but is very limited.) TRANSPORT-ENDTRANSPORT loop is to be used in such a case. It tracks particles with the orbit length variable s as the independent variable as in most accelerator tracking codes, in contrast to PUSH-ENDPUSH loop where the time t is used as the independent variable.

However, the interactions between particles, the interactions with lasers, etc., cannot be handled during the TRANSPORT-ENDTRANSPORT loop. You cannot nest the TRANSPORT-ENDTRANSPORT and the PUSH-ENDPUSH loops in each other.

The purpose of using a pair of commands TRANSPORT and ENDTRANSPORT rather than a single command is similar to the case of PUSH and ENDPUSH, i.e., to allow user's action such as printing during the beamline transport.

The BEAMLINE command does not define the location and direction of the beamline as a whole. You have to define them in TRANSPORT command. Also, you must define the momentum and charge of the reference particle which are needed in converting the bending angles, focal lengths, etc. into the actual magnetic fields.

Syntax:

TRANSPORT BEAMLINE='name', TXYS=(t_0,x_0,y_0,s_0),
  E1=(e_{1x},e_{1y},e_{1s}), E3=(e_{3x},e_{3y},e_{3s}), MOMENTUM=p_0, CHARGE=ϵ
  [RIGHT|LEFT,] [KIND=k|(k_1,k_2), [LOSSMONITOR,] ;

Syntax:

```
ENDTRANSPORT ;
```

name	The beamline name. (You may omit apostrophes.)
t_0, x_0, y_0, s_0	Entrance of the beamline in CAIN coordinate. (m)
E1,E3	Unit vectors along the direction of the horizontal axis and the longitudinal axis of the beamline at its entrance in CAIN coordinate.
p_0	Reference momentum in eV/c.
ϵ	± 1. Sign of the charge of the reference particle.
RIGHT,LEFT	Apply to right- or left-going (in Cain coordinate) particles only. If you do not specify this operand, all the particles having opposite momentum with respect to the beamline direction E3 will be asigned 'lost'. If you specify one of then, RIGHT for example, the left-going ($P_s < 0$ in Cain coordinate) particles are left intact. The right-going ones are transformed to the beamline coordinate and, if $P_s < 0$ in beamline coordinate, they are asigned 'lost'.
KIND	Apply to electrons or positrons only. Of course you must not choose photons.
LOSSMONITOR	Activate loss monitor. Normally, particles going out side the aperture are eliminated from memory. When the loss monitor is activated, they are kept and their coordinates (beamline coordinate) are fronzen where they are lost. You can write their data by WRITE BEAM LOST and plot them by PLOT HIST LOST, etc.

However, the present version simply compares the aperture with the beam (x,y) coordinate only at entrance and exit of magnets. Therfore, the stored position is not exactly on the aperture surface. Also note that in some cases the lost position may be inside the aperture when a particle curls within a magnet and cannot reach the exit of the magnet.

When the beamline consists of n elements, the j-th step ($0 \le j \le n$) is executed as follows.

- At the `TRANSPORT` command in the first step ($j = 0$), the coordinates of the relevant particles are transformed into the beamline coordinate.
- Execute the commands between `TRANSPORT` and `ENDTRANSPORT`. Thus, these commands are repeated ($n + 1$) times.
- At the `ENDTRANSPORT`, the beam goes throughth the ($j + 1$)th element if $j < n$. If $j = n$. the coordinates are transformed back to the CAIN coordinate.

During the loop the predefined parameter `Sbl` contains the current s-coordinate (analogous to `Time` in `PUSH-ENDPUSH` loop). It is zero at $j = 0$ and is the location of the exit of j-th ($1 \le j \le$ number of magnets) magnet. The pre-defined character variable `$PrevMag` is the name of the previous magnet ('`(entr).0`' if $j = 0$) and `$NextMag` is the name of the next magnet ('`(exit).0`' if $j = n$). The name of the magnet is followed by a dot and the order of occurence. For example the second occurene of `QF` is '`QF.2`'. If you want to separate the name and the occurence id, you can say, e.g.,

```
SET n=Strstr($PrevMag,'.'),
    $name=$Substr($PrevMag,1,n-1),
    id=AtoF($Substr($PrevMag,n+1));
```

<u>Caution</u>

- This command is still premature. The accuracy is limited for particles very different from the reference particle (very low energy particle and oppositely charged particles, etc.) Wait for future improvements.
- During the loop the beamline coordinate is used instead of the CAIN coordinate. Thus, the coordinate used in `PRINT` and `PLOT` commands is the beamline coordinate. However, the particles which are excluded by the `RIGHT`,`LEFT` and `KIND` operands are left in the CAIN coordinate.
- You must not use `LORENTZ` command during the loop. Also, you must be careful enough in using `BEAM` command in the loop.

3.21 DO, CYCLE, EXIT, ENDDO

Do loop. Can be nested. Three forms are possible. (`REPEAT` and `WHILE` must be the first operand.)[4]

[4] **CAIN**2.31 introduced the third type `DO i=(...)`. Due to this change, `REPEAT` and `WHILE` are no more positional operands though they should come first. As a result you have to put a comma after `REPEAT/WHILE`.

Syntax: form-1

`DO` `REPEAT,` n ;

n	Number of repetition. Can be an expression (evaluated when entering the loop). $n > 0$. ($n = 0$ causes a jump to `ENDDO`. $n < 0$ causes an abnormal term.)

Syntax: form-2

`DO` `WHILE,` expr ;

expr	A logical expression like in `DO WHILE x>0;`. Any expression is considered to be logical by asigning false for 0 and true otherwise. For example, the do loop `DO WHILE n;` ends when $n = 0$. For compatibility with old versions, The substition operator `=` is treated as `==` in `DO WHILE` (and `IF`) command.

The loop is repeated so long as the condition is satified. The check is made at the time of `DO` command. The values of expressions are REAL*8. If you want integers for definiteness, use `Nint( )` or `Int( )`.

Syntax: form-3

`DO` i=(i_1,i_2[,i_3]);

i	DO control variable. Either a floating scalar variable or an element of a floating array variable. In the former case the variable need not be defined before. In the latter case the array must be declared before. (The subscript is evaluated when entering the DO loop.) Obviously, the variable must not be used as a DO control variable of a lower nest level. (**CAIN** checks this but does not check if the user changes the value.)
i_1,i_2,i_3	The starting value, the upper (lower) bound, and the increment of the control variable, as in FORTRAN. If i_3 is omitted, $i_3 = 1$ is assumed. The value of i_j's are evaluated when entering the DO loop. i_3 must not be zero. If $i_1 = i_2$, the loop is executed once. If $i_1 < i_2$ and $i_3 < 0$ or $i_1 > i_2$ and $i_3 > 0$, the loop is not executed. Don't forget to enclose the numbers by `( )`.

End of do loop is

Syntax:

`ENDDO ;`

Do not forget ";".

As in FORTRAN, `CYCLE` causes a jump to `ENDDO` (at the deepest nest level lower than or equal to the current level) and a return to `DO`. `EXIT` causes a jump to `ENDDO` and the end of the loop. For example,

```
  DO i=1,5;
```

```
    DO j=1,10;
      IF x>0;
        IF y>0; EXIT; ENDIF;
      ENDIF;
    ENDDO;
  ENDDO;
```
will cause the end of the j-do loop but the i loop still continues. A jump getting out of `PUSH` or `TRANSPORT` loop is prohibited.

3.22 IF, ELSEIF, ELSE, ENDIF

Define if block. Can be nested. Note that 'THEN' is not needed. The `ELSEIF` and `ELSE` clause may be absent.

Syntax:

```
IF      expr ;
. . . . . . . . . . . . ;
ELSEIF      expr ;
. . . . . . . . . . . . ;
ELSE ;
. . . . . . . . . . . . ;
ENDIF ;
```

expr	A logical expression like in `IF x>0;`. Any expression is considered to be logical by asigning false for 0 and true otherwise. For example, the IF statement `IF n;` cause a jump to `ELSEIF;` or `ELSE;` when $n \neq 0$. For compatibility with old versions, the substition operator `=` is treated as `==` in `IF` and `ELSEIF` (and `DO WHILE`) command.

Do not forget ";".

3.23 WRITE, PRINT

Write some data. The only difference between `WRITE` and `PRINT` is the default destination which is `OutFile` for `WRITE` and `MsgFile` for `PRINT`. Therefore, they are identical if `FILE` operand is specified. Another difference is that errors in reading the command cause abnormal termination for `WRITE` whereas the command is ignored for `PRINT`.

Following is the list of possible forms of `WRITE` commands. (`WRITE` can be replaced by `PRINT`).

`WRITE BEAM`	Write individual data of the particles. Sec.3.23.1
`WRITE STATISTICS`	Write statistical data of the beam (e.g., beam size). Sec.3.23.2
`WRITE LUMINOSITY`	Write luminosity already computed. Sec.3.23.3
`WRITE MAGNETS`	Write list of defined magnets. Sec.3.23.4

`WRITE BLOPTICS`	Write linear optics of a beamline. Sec.3.23.5
`WRITE BLGEOMETRY`	Write the geometry of a beamline. Sec.3.23.6
`WRITE PARAMETER`	Write values of expressions. Sec.3.23.7
`WRITE ARRAY`	Write list of allocated arrays. Sec.3.23.8
`WRITE CPUTIME`	Write the cpu time. Sec.3.23.9

3.23.1 Write the macro-particle data

Syntax:

`WRITE BEAM,` [`FILE=`f_n|`'`file_name`'`,] [`APPEND,`] [`SHORT`|`MATHEMATICA`]
[`RIGHT`|`LEFT,`] [`KIND=`k`,`] [`INCP,`] [`LOST,`] [`SELECT=`f_{sel}`,`] `;`

`BEAM` Write beam data.

f_n,file_name,`APPEND` Unit number or name of an old or new file. For the differece between the two forms, See Sec.2.8.

`SHORT` Short format which (may) fits to a monitor screen.

`METHEMATICA` MATHEMATICA list style format. Use standard format (see Sec.3.5) if none of the above two are specified.
Caution: The list style in the output file is
{. . .}, {. . .}, . . ., {. . .},
i.e., you need to add { at the top of the file and, at the end, replace the last comma by } to get complete MATHEMATICA list format.

`RIGHT`|`LEFT` Write only either right-going or left-going particles. Default=both.

`INCP` Write particles created by incoherent processes (defined by `PPINT` command). Otherwise, normal particles only. If you want both, execute the command twice.

`LOST` Write particles lost in beamlines. (See Sec.3.20) Only lost particles are written. Otherwise, lost particles are not written.

k Write only photon ($k = 1$) or electron ($k = 2$) or positron ($k = 3$) selectively. Default=all.

f_{sel} Logical function for selecting particles, e.g.,
`SELECT=( X>0 )` will select particles with positive x coordinate. See Sec.3.31 for more detail.

3.23.2 Write the beam statistics data

Syntax:

`PRINT STATISTICS,` [`INCP,`] [`SHORT`|`LONG,`] [`LOST,`]
[`FILE=`f_n|`'`file_name`'`,] [`APPEND,`] `;`

STATISTICS	Write beam global data such as number of particles, r.m.s. size, etc.
SHORT	Print only the number of macro- and real particles. If none of SHORT and LONG is specified, print average and r. m. s. of (t, x, y, s) and (E, p_x, p_y, p_s) as well as the average spin components.
LONG	Print max. and min. in addition to the standard items.
INCP	Include incoherent particles only. Otherwise, normal particles only. If you want both, execute the command twice.
LOST	Include particles lost in beamlines. (See Sec.3.20) Only lost particles are included. Otherwise, lost particles are not included.
f_n,file_name,APPEND	Unit number or name of an old or new file. For the differece between the two forms, See Sec.2.8.

3.23.3 Write the calculated luminosity

Syntax:

```
PRINT    LUMINOSITY,  KIND=(k1,k2)   [FILE=fn|'file_name',]   [APPEND,]
;
```

LUMINOSITY	Write calculated luminosity specified by (k_1,k_2).
k_1,k_2	Define right and left-going beams. All the luminosities (differential and polarization) defined by the LUMINOSITY command will be printed. The print format is complicated. Just try.
f_n,file_name,APPEND	Unit number or name of an old or new file. For the differece between the two forms, See Sec.2.8.

3.23.4 Write a list of defined magnets

Syntax:

```
WRITE    MAGNETS,  [BEAMLINE='name',]   [COMBINE,]   [MOMENTUM=p0,]
  [FILE=fn|'file_name',]   [APPEND,] ;
```

name	Name of a defined beamline. The apostrophes can be omitted. If the beamline name is not given, write a list of all defined magnets. If given, only those which actually appear in the beam line with the number of appearance are listed.
COMBINE	Same name magnets with zero length drift spaces in between are to be listed as one magnet. Effective when the beamline name is specified.
p_0	Reference momentum in eV/c. If given, print also the field strength in Tesla or Tesla/m.

3.23.5 Write the beamline optics

Syntax:

> WRITE BLOPTICS, BEAMLINE='name', [FILE=f_n|'file_name',] [APPEND,] ;

name	Name of a defined beamline. The apostrophes can be omitted. The optics must be calculated by using `BLOPTICS` command in advance.

Caution: You get an error message when you have not called the `BLOPTICS` nor the `MATCHING` command. However, if you changed magnet parameters and have not called the `BLOPTICS` command since then, `WRITE BLOPTICS` command will write down a wrong data without error message.

3.23.6 Write the beamline geometry

Syntax:

> WRITE BLGEOMETRY, BEAMLINE='name', TXYS=(t_0,x_0,y_0,s_0), E3=(e_x,e_y,e_s), E1=(r_x,r_y,r_s), [FILE=f_n|'file_name',] [APPEND,] ;

name	Name of a defined beamline. The apostrophes can be omitted.
(t_0, x_0, y_0, s_0)	Location of the beamline entrance in CAIN coordinate. Actually t_0 is not used but it is retained for uniformity (and for possible future use).
(e_x, e_y, e_s)	Unit vector (in CAIN coordinate) along the orbit direction of the beamline at the entrance. This defines the s-axis of the beamline.
(r_x, r_y, r_s)	Unit vector (in CAIN coordinate) along the radial direction of the beamline at the entrance. This defines the x-axis of the beamline.

3.23.7 Write the values of parameters and expressions

Syntax:

> PRINT [PARAMETER,] [FILE=f_n|'file_name',] x_1[, x_2[, x_3...]], [FORMAT=(fmt),|FORMAT=fmt,] ;

PARAMETER	Write values of (predefined or user defined) parameters or expressions. Can be omitted.
f_n	File reference number. See above for default.
x_j	Expressions. It is safer to enclose each expression by `( )` or `[ ]` or `{ }`.[5] It is also possible to write a do-type sequence of the form (almost like

[5] There is no such a rule that a user parameter name must not be identical to some keyword. Here, however, there is an inconsistency of grammer. If you define a parameter with the name `ST`, for example, `PRINT ST` may be understood as printing the parameter or printing the statistics, unless the keyword `PARAMETER` is explicitly written. This can be avoided by writing `PRINT (ST)` because `(ST)` is not a keyword but is an expression actually identical to `ST`.

FORTRAN)
`(`y_1`,...,`y_n`,i=`i_1`,`i_2`,`i_3`)`
where y_j's are expressions, i is a floating type user-parameter name (need not be defined by `SET` command) or an element of floating type array, i_1, i_2, and i_3 are expressions for initial, final, and increment values of i. If i_3 is omitted, $i_3 = 1$ is adopted. Note that i_1, i_2, and i_3 are considered to be integers. (`Nint` is applied.)
Do-type sequence may be nested as in FORTRAN. The do-control variable must not duplicate, of course. (Duplication within the sequence and the loop of `DO` command is checked but possible interference with variables outside `PRINT` or `WRITE` is not checked.)

`(`fmt`)`,fmt Fortran format. Character expression or a character string enclosed by `( )`. The latter is for backward compatibility.[6]
CAIN recognizes the following format descriptors:
`/ X T P A I F E D G`
When I-format is called for, `Nint` is applied for the corresponding argument.[7] If format is not specified, printed as 'expression=value' by `(1PD15.8)` for floating or by `('"',A,'"')` for character type (one line for each). An exception is that the value is just printed by `(A)` without 'expression=' when the expression is a character type literal constant. Thus, `PRINT 'abc'` will cause `abc` be printed.
If format is given but there is no expression to be printed, the format is executed as in FORTRAN. For example,
`WRITE FORMAT=('nothing');`
will cause '`nothing`' be printed.[8]
Unfortunately, the grammer of **CAIN** does not allow an un-paired apostrophe so that, for example, `1H'` will cause an error.

f_n,file_name,`APPEND` Unit number or name of an old or new file. For the differece between the two forms, See Sec.2.8.

3.23.8 Write a list of all allocated arrays

Syntax:

`PRINT` `ARRAY,` [`LONG,`] [`FILE=`f_n|`'`file_name`',`] `;`

`LONG` Write the values of array contents. Otherwise, only a list of array names with their sizes are printed.

[6]Here is a problem of grammer. Only the character expression should be accepted for consistency of grammer, but a string with `( )` should be retained for backward compatibility. A problem arises e.g., for `FORMAT=('(I)')` which can be interpreted in both ways, print three characters `(I)` if interpreted in the old way or print something with I-format in new way. **CAIN** assumes the old form at first.

[7]I-format was not allowed before **CAIN**2.3.

[8] A known bug. `WRITE (i=1,2), FORMAT=('nothing');` won't work. Instead, you can write `WRITE ('nothing',i=1,2);`

3.23.9 Write the cpu time

Print the cpu time since the job start.
Syntax:

```
PRINT    CPUTIME, [LONG] ;
```

`LONG`	Print cpu time in major subroutines. Otherwise, the total only.

3.24 PLOT

Plot using TopDrawer.
Following is the list of possible forms of `PLOT` commands.

`PLOT HISTOGRAM`	Histogram of particles. Sec.3.24.1
`PLOT SCATTER`	Scatter plot of particles or laser photons. Sec.3.24.2
`PLOT TSTPARTICLE`	Plot test particle data. Sec.3.24.3
`PLOT LUMINOSITY`	Differential luminosity. Sec.3.24.4
`PLOT BBFIELD`	Charge distribution and beam field. Sec.3.24.5
`PLOT BLOPTICS`	Beamline optics (Twiss paramewters). Sec.3.24.6
`PLOT BLGEOMETRY`	Beamline geometry. Sec.3.24.7
`PLOT FUNCTION`	Function expressed by ‘expression’. Sec.3.24.8

3.24.1 Histogram of particle data

Syntax:

PLOT HISTOGRAM, [NONEWPAGE,] [RIGHT|LEFT,] [KIND=k|(k_1,k_2),]
[INCP,] [LOST,] [SELECT=f_{sel},] H=f_x, [HSCALE=(x_{min},x_{max},n_{bin}),]
[HLOG,|HLINEAR,] [VLOG,|VLINEAR,] [COLOR=color,]
[TITLE='head_title',] [HTITLE='bottom_title',] [VTITLE='left_title',]
[FILE=f_n|'file_name',] [APPEND,] ;

`NONEWPAGE` Do not insert ‘NEWFRAME’ of TopDrawer so that the figure is written on the previous plot on the same file. This makes sense when the new plot has the same scale as the previous plot.
The `FILE` operand is ignored. The default values of the following parameters are those in the previous plot so that they need not be specified if the same values are to be used (this list includes parameters for `PLOT SCATTER` etc.): (`RIGHT`,`LEFT`), `KIND`, `SELECT`, `FILE`, `MAXNP`, `COLOR` `H`, `V`, `HSCALE`, `VSCALE`, (`HLOG`,`HLINEAR`), (`VLOG`,`VLINEAR`)
The following parameters are not inherited from the previous plot: (`HISTOGRAM`,`SCATTER`,etc), `INCP`, `LOST`, `TITLE`, `HTITLE`, `VTITLE`
A problem is the vertical scale for histogram which is determined from the data contents. It may not work as you want.

RIGHT|LEFT Select right(left)-going particles only.

k,k_1,k_2 Select photons ($k = 1$), electrons ($k = 2$), positrons ($k = 3$) only.

INCP Include particles created by incoherent incoherent processes only. Otherwise normal particles only.

LOST Include particles lost in beamlines. (See Sec.3.20) Only lost particles are included. Otherwise, lost particles are not included.

f_{sel} Logical function for selecting particles, e.g.,
`SELECT=( X>0 )` will select particles with positive x coordinate. See Sec.3.31 for more detail. Note that, once `SELECT` is specified, both normal and incoherent particles are included by default. If you reject incoherent particles, you must say `PLOT ..., SELECT=(Incp==0);`.

f_x An expression defining the horizontal variable. Following running variables can be used. (See Sec.2.5)
`T`, `X`, `Y`, `S`, `En`, `Px`, `Py`, `Ps`, `Sx`, `Sy`, `Ss`, `Xi1`, `Xi2`, `Xi3`, `Kind`, `Gen`, `Wgt`
For example,
`H=Sqrt[(Px^2+Py^2)/Ps^2]*1E6`
defines the orbit angle in micro-radians.

x_{min},x_{max},n_{bin} Minimum and maximum of the horizontal scale and the number of bins. If omitted, the minimum and maximum in the particle data are used for x_{min},x_{max} and $n_{bin} = 50$.

HLOG,VLOG Log scale of horizontal and vertical axes. When `HLOG` is specified, x_{min} and x_{max} must be specified explicitly. The binning interval will be equal in log-scale.

color Color name allowed in TopDrawer. One of the following:
`WHITE`, `RED`, `YELLOW`, `GREEN`, `CYAN`, `BLUE`, `MAGENTA`, `BLACK`
Note that `WHITE` appears black and `BLACK` is invisible.

top_title, etc Title string. Must be enclosed by a pair of apostrophes. Topdrawer case string can be specified by using “`;`” as the delimitor like
`TITLE='EOG1; XGX;'`, for writing E_γ. It is recommended to put “`;`” also at the end, as in this example, to avoid writing unnecessary blanck characters.[9]

f_n,file_name,APPEND Output file unit number or name of an old or new file. If not specified, `FILE=TDfile`. For the differece between the two forms, See Sec.2.8.

[9] There is a subtle problem related to the quote `'` and the double quote `"`. For example, **CAIN** recognizes `"'"` as a character `'` but, since TopDrawer does not understand `"` so that it is transformed to `'''`. If you want one `'`, you have to write `"''"`, which bocomes `''''` in the TopDrawer input data and is understand as one single `'` by TopDrawer.

3.24.2 Scatter plot of particles or laser photons

Syntax:

> `PLOT SCATTER,` `[NONEWPAGE,]` `[RIGHT|LEFT,]` `[KIND=`k`|(`k_1`,`k_2`),]` `[INCP,]` `[LOST,]` `[SELECT=`f_{sel}`,]` `H=`f_x`,` `V=`f_y`,` `[HSCALE=(`x_{min}`,`x_{max}`),]` `[VSCALE=(`y_{min}`,`y_{max}`),]` `[HLOG,|HLINEAR,]` `[VLOG,|VLINEAR,]` `[COLOR=`color`,]` `[MAXNP=`n_{max}`,]` `[TITLE='`top_title`',]` `[HTITLE='`bottom_title`',]` `[VTITLE='`left_title`',]` `[FILE=`f_n`|'`file_name`',]` `[APPEND,]` `;`

f_y	An expression defining the vertical variable.
y_{min},y_{max}	Minimum and maximum of the vertical scale.
n_{max}	Maximum number of points to be plotted (in order to save the plotting time). Randomly selected. Default: plot all points.

Other operands are the same as for the histogram.

You can also plot laser intensity profile as a scatter plot of laser photons. Unfortunately, there is no way to select a laser if more than one lasers are defined.

Syntax:

> `PLOT SCATTER,` `LASERPHOTON,` `[NONEWPAGE,]` `T=`t`|S=`s`,` `[MAXNP=`n`,]` `H=`f_x`,` `V=`f_y`,` `[HSCALE=(`x_{min}`,`x_{max}`),]` `[VSCALE=(`y_{min}`,`y_{max}`),]` `[HLOG,|HLINEAR,]` `[VLOG,|VLINEAR,]` `[COLOR=`color`,]` `[TITLE='`top_title`',]` `[HTITLE='`bottom_title`',]` `[VTITLE='`left_title`',]` `[FILE=`f_n`|'`file_name`',]` `[APPEND,]` `;`

t,s	Either one of these must be specified. (meter). Snapshot if t is specified, race-goal shot if s is specified.
n	Number of laser photons to be plotted. (default=1000)

Other operands are the same as for the scatter plot of particles.

The following example (to be inserted during a `PUSH-ENDPUSH` loop) will cause a (s,x) snapshot of laser and electron beams in the same frame:

```
PLOT SCAT, LASER, T=Time, COLOR=RED, H=S/1e-3, V=X/1e-6,
     HTITLE='S (mm);', VTITLE='X (Mm);   G  ;',
     TITLE='t='+$FtoA(Time/1e-3,'(F7.2)')+'mm;';
PLOT SCAT, NONEWPAGE, KIND=2, COLOR='WHITE', MAXNP=0;
```

Note that `Time` is the running variable during a `PUSH-ENDPUSH` loop.

3.24.3 Plot the test particle data

Syntax:

PLOT TESTPARTICLE, [RIGHT|LEFT,] [KIND=k|(k_1,k_2),] H=f_x, V=f_y, [HSCALE=(x_{min},x_{max}),] [VSCALE=(y_{min},y_{max}),] [COLOR=color,] [TITLE='top_title',] [HTITLE='bottom_title',] [VTITLE='left_title',] [FILE=f_n|'file_name',] [APPEND,] ;

Other operands are the same as for the scatter plot. Note that the information of the test particle history is stored (in contrast to normal particles). Thus, you can say, for example, `H=T` to see the trajectory as a function of time.

The plot may show apparently unphysical features when you apply `DRIFT` command. `DRIFT` command may be used to pull particles to a certain position or time. This does not corresponds to a physical motion. Even in such cases, test particle coordinates are stored at the end of `DRIFT` command. Moreover, in contrast to the `PUSH` command, step-by-step information of test particles during `DRIFT` command is not stored because `DRIFT` command calculates particle trajectories by a single step using exact analytic formulas.

3.24.4 Plot the differential luminosity

The differential luminosity w.r.t. the center-of-mass energy can be plotted if defined by `LUMINOSITY` command and calculated by `PUSH` command. Only the 1-D differential luminosity $d\mathcal{L}/dW$ is plotted. 2-D differential luminosity $d\mathcal{L}/dE_1dE_2$ is not plotted because the TopDrawer available at KEK HP station is not capable of 3-D plot.

Syntax:

PLOT LUMINOSITY, KIND=(k_1,k_2), [FILE=f_n|'file_name',] [APPEND,] [VLOG,|VLINEAR,] [PERBIN|PERHVAR,] [COLOR=color,] ;

k_1,k_2	Define right and left-going beams. When `HELICITY` operand has been specified in the `LUMINOSITY` command, all the 5 spectrums (unpolarized and 4 combinations of helicities) come out in 5 separate plots.
VLOG	Log scale of vertical axis. The horizontal axis cannot be log-scale.
PERBIN	Luminosity per bin (1/cm^2/sec/bin) is plotted.
PERHVAR	Luminosity per unit increment of horizontal axis (energy) is plotted. (1/cm^2/sec/eV). Default is `PERBIN`.

More flexible plot is possible with the complicated syntax

Syntax:

PLOT LUMINOSITY, KIND=(k_1,k_2), V=f, [NONEWPAGE,] [VSCALE=(y_{min},y_{max}),] [VLOG,|VLINEAR,] [PERBIN|PERHVAR,] [COLOR=color,] [TITLE='top_title',] [HTITLE='bottom_title',] [VTITLE='left_title',] [FILE=f_n|'file_name',] [APPEND,] ;

f	Defines what is plotted. You can use the following variables. `L0`: unpolarized luminosity. `Ln`: n=1,2,3,4. helicity luminosity. `Lij`: i,j=0,1,2,3. general polarization luminosity. These are in units of $1/\text{cm}^2/\text{s}/\text{bin}$. (Or $1/\text{cm}^2/\text{s}/\text{eV}$ if `PERHVAR` is specified.) `Ln` (`Lij`) is allowed when `HELICITY` (`ALLPOL`) has been specified in `LUMINOSITY` command.

The operands `KIND`, `PERBIN`, `PERHVAR` are the same as in the first syntax. The rest is the same as in `PLOT SCATTER` except for `V=`f. The titles are automatically created in the first syntax but not in the second.

3.24.5 Plot charge distribution and beam-beam field

The charge distribution and the beam field data for beam-beam interaction are computed at each time step for each longitudinal slice but they are not kept in the memory. They can be plotted only at the time moment and for the slice which is being proccessed. Thus, this command is to be inserted during `PUSH` loop. The slice is specified by the `S` operand. Syntax:

```
PLOT    BBFIELD,   S=s1|S=(s1,s2,...),    [FILE=fn|'file_name',]
  [APPEND,] ;
```

s_j	Define the s-coordinate. Plot for the slice which contains one of s_j's. Upto 5 s_j's can be specified.

3.24.6 Plot beamline optics

Syntax:

```
PLOT    BLOPTICS,   BEAMLINE='name',    [INTERPOLATE=Δs,]
  [FILE=fn|'file_name',]    [APPEND,] ;
```

name	Name of a defined beamline. The apostrophes can be omitted. The optics must be calculated by using `BLOPTICS` command in advance.
Δs	Interpolate optics functions so that the step size $\leq \Delta s$ (meter). Otherwise, the values at magnet borders are linked by straight lines.

Caution: You get an error message when you have not called the `BLOPTICS` command. However, if you changed magnet parameters and have not called the `BLOPTICS` command since then, `PLOT BLOPTICS` command will will write down a wrong data without error message.

3.24.7 Plot beamline geometry

Syntax:

> `PLOT BLGEOMETRY, BEAMLINE='name', [TXYS=(`t_0, x_0, y_0, s_0`),]`
> `[E3=(`e_x, e_y, e_s`),] [E1=(`r_x, r_y, r_s`),] [VHRATIO=`r_{vh}`,] [MAGWIDTH=`w`,]`
> `[PAPER=(`w_h, w_v`),] [FILTER='f',] [FILE=`f_n`|'file_name',] [APPEND,] ;`

name	Name of a defined beamline. The apostrophes can be omitted. The optics must be calculated by using `BLOPTICS` command in advance.
(t_0, x_0, y_0, s_0)	Location of the beamline entrance in CAIN coordinate. Actually t_0 is not used but it is retained for uniformity (and for possible future use). Default is $(0, 0, 0, 0)$.
(e_x, e_y, e_s)	Unit vector (in CAIN coordinate) along the orbit direction of the beamline at the entrance. This defines the s-axis of the beamline. Default is $(0, 0, 1)$.
(r_x, r_y, r_s)	Unit vector (in CAIN coordinate) along the radial direction of the beamline at the entrance. This defines the x-axis of the beamline. Default is $(1, 0, 0)$.
r_{vh}	Change the vertical scale in the plot with respect to horizontal. When $r_{vh} > 1$, vertical scale is magnified. Default=1.
w	Magnet (full) width in meters. Dipole magnets appear as a box of width w and quadrupoles as $0.75w$. Default=1.0.
(w_h, w_v)	Paper size in inches. Default=(13,10).
f	Character string of the filter of the magnet names to be printed. It may contain the wild card '`*`'. (e.g., `FILTER='B*'` will print names starting with 'B' only.) If not specified, names of all the dipoles and quadrupoles are printed.

3.24.8 Plot a function

Syntax:

> `PLOT FUNCTION, [NONEWPAGE,] H=`f_x`, V=`f_y`, PARAMETER=name,`
> `RANGE=(`x_1`,`x_2`[,`n`]), [XLOG,|XLINEAR,] [HSCALE=(`x_{min}`,`x_{max}`),]`
> `[VSCALE=(`y_{min}`,`y_{max}`),] [HLOG,|HLINEAR,] [VLOG,|HLINEAR,]`
> `[LINEMODE=(`l_1`,`l_2`,,,...),] [COLOR=color,] [TITLE='top_title',]`
> `[HTITLE='bottom_title',] [VTITLE='left_title',] [FILE=`f_n`|'file_name',]`
> `[APPEND,] ;`

f_x,f_y	Define the function to be plotted in the parameterized form. They should normally contain the variable defined by the `PARAMETER` command.

name	Name of the parameter to vary. Must satisfy the constraints as a user-defined variable and must not be the pre-defined names. If you want to plot a circle, for example, you would say `PLOT FUNCTION, PARAMETER=t, RANGE=(0,2*Pi),` `H=Cos(t), V=Sin(t), HSCALE=(-1.5,1.5), VSCALE=(-1.2,1.2) ;`
x_1,x_2	Define the range of the parameter. ($x_1 \neq x_2$)
n	Number of points (-1) in the range (x_1, x_2). (Default $n = 100$.)
XLOG	Divide the range uniformly in log scale. Otherwise linear.
l_1,l_2,...	Define the line mode, meaning a line segment of length l_1 (in units of inches) followed by a space of length l_2, followed by a line l_3, etc. The whole pattern is repeated. For example, `LINEMODE = (0.1,0.1)` will cause a dashed line. If `LINEMODE` is not defined or only l_1 is specified, a solid line is plotted.

Other operands are the same as for the histogram. You can plot many functions in a frame by using `NONEWPAGE` option.

3.25 CLEAR

Clear/disable the beam, laser, etc. The first operand is a positional keyword, one of `BEAM`, `LASER`, `LASERQED`, `LUMINOSITY`, `BBFIELD`, `EXTERNALFIELD`, `CFQED`, `PPINT`.

Clear particles

Syntax:

CLEAR BEAM[,]† [TESTPARTICLE,] [INCP,] [RIGHT|LEFT,]
[KIND=k|(k_1,k_2),] [SELECT=f_{sel},] ;

TESTPARTICLE	Clear test particles.
INCP	Clear particles created by incoherent processes (defined by `PPINT`). If none of `TESTPARTICLE` and `INCP` is specified, normal particles are eliminated. Therefore, if you want to eliminate all, you need `CLEAR` command twice: `CLEAR BEAM, TESTPARTICLE, INCP;` `CLEAR BEAM;` or use `SELECT` operand: `CLEAR BEAM, SELECT=( )`
RIGHT,LEFT	Clear right or left-going particles only. (Default=both).
k,k_1,k_2	Clear photon ($k = 1$) or electron (2) or positron (3) only.
f_{sel}	Logical function for selecting particles. See Sec.3.31 for detail.

Turn off lasers

Syntax:

```
CLEAR    LASER[,]† ;
```

Turn off `LASERQED`

Syntax:

```
CLEAR    LASERQED[,]†   [COMPTON,]   [BREITWHEELER,] ;
```

Clear parameters for the laser QED. If either one of `COMPTON` or `BREITWHEELER` is specified, the other one is not turned off.

Clear luminosity

Syntax:

```
CLEAR    LUMINOSITY[,]† ;
```

Clear luminosity integrals as well as the definitions of luminosities. Note that the contents of luminosity integrals are cleared whenever the `PUSH` command starts. Thus, if you do another `PUSH` without `CLEAR LUMINOSITY`, the luminosity command will be still active and the integration starts from scratch..

Turn off beam-beam field

Syntax:

```
CLEAR    BBFIELD[,]† ;
```

Turn off the external field

Syntax:

```
CLEAR    EXTERNALFIELD[,]† ;
```

Turn off `CFQED`

Syntax:

```
CLEAR    CFQED[,]†   [BEAMSTRAHLUNG,]   [COHERENTPAIR,] ;
```

If none of `BEAMSTRAHLUNG` and `COHERENTPAIR` is specified, both is turned off.

Turn off `PPINT`

Syntax:

```
CLEAR    PPINT[,]† ;
```

Turn of particle-particle interaction defined by `PPINT` command. Note that this does not mean to eliminate particles already created.

3.26 FILE

Open, close, rewind a file. The first operand is positional.
Syntax:

> FILE   OPEN|CLOSE|REWIND|ENDFILE[,]† UNIT=n_f,   [STATUS='status',]
>   [NAME='fname',] ;

n_f	Logical file number. No default.
status	For OPEN, one of `NEW`, `OLD`, `SCRATCH`, `UNKNOWN`. Default=`UNKNOWN`. For CLOSE, one of `KEEP`, `DELETE`. Default=`KEEP`. Not used for REWIND and ENDFILE. Need not be enclosed by apostrophes.
fname	File name. Used for OPEN only.

3.27 HEADER

Define the header for TopDrawer plots. Effective until next `HEADER` command appears.
Syntax:

> HEADER   'header_string' ;

header_string	Character string. Upto 120 characters. Strings delimited by commas like 'string$_1$','string$_2$',... are concatenated. The 'case' string for TopDrawer can be defined by using semicolon ";" as delimiter, like `HEADER 'JLC ECM1=500GeV;      X   X         ' ;` If header_string is not written, the header is cleared.

3.28 STORE and RESTORE

Store/restore the current variables or the luminosity data in/from a file. The latter is needed when you compute the luminosity in a run and print/plot it in the next run.
Syntax:

> STORE   [LUMINOSITY,]   [FILE=f_n|'fname',] ;

Syntax:

> RESTORE   [LUMINOSITY,]   [FILE=f_n|'fname',] ;

LUMINOSITY	Store/restore luminosity data. If not specified, store/restore variables.
f_n, fname	File unit number or file name. When a file name is specified, the file is opened with the unit number 98 and is closed after reading/writing. When the unit number is given, no process of open and close is done. If either is omitted, the standard name (`'stdstfl.dat'` for variables and `'stdstolum.dat'` for luminosity both in `exec` directory) is used. The

file is written in ascii format but do not try to edit it. No protection against wrong formats.

When `STORE` is called, all the variables are written in a file.[10] At the time of `RESTORE` there can be three kinds of variables (except for unchangeable ones): those already defined and appear in the file, already defined but do not appear in the file, and undefined variables. The first kind variables are overwritten. The second ones are kept (not eliminated) and the last ones are added.

When `STORE LUMINOSITY` is called, all the luminosity data at that time will be written in the specified file. When `RESTORE LUMINOSITY` is invoked, all the luminosity data in the present run is erased and replaced by the data in the file.

These two commands are introduced for convenience in splitting a job into two jobs for calculation and for output. For example, if you expect a long job but do not know what is to be printed/plotted. You write

```
WRITE  BEAM,  FILE='beam_file';
STORE;
STORE LUMINOSITY;
```

near the end of the long job. Then, you can print/plot the beam in the next job by

```
BEAM   FILE='beam_file';
RESTORE;
RESTORE LUMINOSITY;
PLOT  .........
```

Here, in the `PLOT` command you can use the variables you defined in the previous job. If you want different plots, you can repeat the second job.

However, keep in mind that these commands are not intended to split a job at arbitrary point. Only the user variables, luminosity data and particle data can be transfered to later jobs by `STORE`, `RESTORE`, and `WRITE BEAM` commands.

3.29 STOP

Stop **CAIN** run.

3.30 END

Indicates the end of input data. If absent, added at the end of file. At the beginning of **CAIN** run, the input file is read through until `END` (or end-of-file) and the command structure (command names, the terminator "`;`", nest of `DO`/`IF`/`PUSH`/`TRANSPORT`) is checked. Thus, the grammer beyond `END` is not checked in contrast to `STOP`.

[10] Exceptions are the two variables `MsgFile` and `MsgLevel`. They are not stored and, hence, not restored.

3.31 Particle selection operand

Several commands such as `PLOT` allow to select particles by using the keywords like `RIGHT`, `LEFT`, `KIND`. A more powerful way is to use `SELECT` with the operand syntax `SELECT=`f_{sel}.

The function f_{sel} is any logical expression involving particle properties (coordinate, energy-momentum, etc). If it is zero (false), the particle is not selected. The particle variables that you can use in this context is

`T,X,Y,S`	particle time-space coordinate (m).
`En,Px,Py,Ps`	energy-momentum (eV, eV/c).
`Sx,Sy,Ss`	Electron/positron spin.
`Xi1,Xi2,Xi3`	Photon Stokes parameters ξ_1, ξ_2, ξ_3.
`Kind`	Particle species. 1,2,3 for photon, electron, positron.
`Gen`	Particle generation.
`Wgt`	Macro-particle weight. One macro-particle corresponds to `Wgt` real particles.
`Incp`	Logical variable. True (=1) if the particle is created by an incoherent process. False (=0) otherwise.

For examples,

```
SELECT= ( Kind==2 && En> 100e9 && Sqrt(Px^2+Py^2)>1e9 )
```

for selecting electrons with energy over 100GeV and transverse momentum over 1GeV/c. The outermost parenthesis are used for clarity. They are not needed grammatically.

Thus, the keywords `RIGHT`, `LEFT` and `KIND` are not needed. (`RIGHT` can be replaced by `SELECT=(Ps>0)`.) However, they are still retained and recommended.[11] These keywords define the subset of particles to apply `SELECT` operand. Another selecting keyword `LOST` cannot be replaced by `SELECT`.

There was a keyword `GENERATION` (or `GEN`) in old versions of **CAIN** for selecting particles with specified generation. This keyword is not used since **CAIN**2.3 because its syntax like `GEN>2` is not compatible with the new grammer. You can now say `SELECT= Gen>2`.

There is a complication on the selecting operand `INCP`, which exclusively select incoherent particles. The versions since **CAIN**2.32 apply the following rule for lower compatibility of input data. In the commands like `PLOT`,

- When `INCP` is specified, include only incoherent particles.
- When `SELECT` is specified, include both normal and incoherent particles.
- When none of `INCP` and `SELECT` is specified, include only normal particles.
- In any case `SELECT`, if specified, applies on the particle subset defined above.

[11] The evaluation of the expression f_{sel} is done by a sort of 'compiled load module' rather than 'interpretator', but still the computing time is a problem.

For example, if you want to plot right-going normal particles, you say
`PLOT HIST, RIGHT,.....`
or, you can also say
`PLOT HIST, SELECT=('Ps>0 && Incp==0'),......`
Don't miss `Incp==0` in the latter case if your beam contains incoherent particles. If you want on the other hand to plot right-going incoherent particles, you say
`PLOT HIST, RIGHT, INCP, .....` or, `PLOT HIST, SELECT=('Ps>0 && Incp==1'),..`

A similar rule applies to the beam statistics functions. For example,

`AvrEn(1,1)`	average energy of normal, right-going photons
`AvrEn(1,1,'Incp==1')`	average energy of incoherent, right-going photons
`AvrEn(1,1,' ')`	average energy of all right-going photons.

Chapter 4

Installation

The development platform of **CAIN** used to be a UNIX machine but since **CAIN**2.2 it moved to Windows. There has been inconvenience for UNIX users. **CAIN**2.35 now tries to support UNIX version too although there may still be several problems depending on the species of UNIX.

4.1 UNIX Version

To obtain **CAIN** by anonymous ftp

1. Go to the **CAIN** home page `http://www-acc-theory.kek.jp/members/cain/` and click at `cain235.tar.gz`. Then you get `cain235.tar.gz`. (about 375kB).
2. Or directly to the ftp site `ftp://lcdev.kek.jp/pub/Yokoya/cain235/cain235.tar.gz`
3. '`gunzip`' it, you get `cain235.tar`. It is a '`tar`'ed file. Move it to an appropriate directory.
4. 'untar' it by `tar -xvf cain235.tar`
   Then, a new directory `cain235` (overwritten if already exists) will be created under the current directory. You may rename the root directory `cain235` (but not the subdirectories). In the following we assume the root directory is `cain`. This new directory contains five subdirectories `exec`, `in`, `out`, `src`, `doc`.[1] The directory `doc` contains a file `readme.txt`.

You can also download the manual you are reading from the same home page (gzip'ed postscript file: manual-cain235.ps.zip). [2]

4.1.1 Directory Structure

The directory structure is shown below.

[1] There may be one more directory `out0` containing the outputs from example data.
[2] Or from the ftp site `ftp://lcdev.kek.jp/pub/Yokoya/manual-cain235.ps.zip`

```
cain ┬─src ┬─many files  most of the fortran source files.
     │     ├─unix        source files for unix (actually, dummy routines of Windows)
     │     ├─module      source files containing MODULEs.
     │     ├─include     all the files to be INCLUDEd.
     │     └─obj         directory to store objects
     ├─exec               load module, shell scripts
     ├─in                 input data
     ├─out                outputs
     └─doc                documents
```

4.1.2 Compilation

Since **CAIN**2.2 FORTRAN90 has been adopted. This may be inconvenient to UNIX users (no free compiler) but it is necessary for using dynamic allocation by standard FORTRAN statements. (Since only the binary is distributed for Windows version, users cannot re-compile **CAIN** with changed array dimension.)

Following steps are needed for compiling **CAIN**.

1. If you do not have FORTRAN90 compiler, you have to buy one.
2. You may have to change all the file names `*.f` in `src/`, `src/unix` and `src/module` to `*.f90`, depending on your compiler.
3. Compile `src/module/flchtype.f` first.
4. Then compile other `*.f` files in `src/module/`.
5. Compile all files in `src/unix/` and `src/`.
6. Link all.

The directory `cain/src/` contains `Makefile` for make. It works on the SAD computer at KEK (Compaq Tru64, compiler:Compaq Fortran 90). The file is written assuming the following rule.

- This make is to be executed at `cain/src/`.
- All object files (`*.o`) are to be stored in `cain/src/obj/`.
- All module files (`*.mod`) are to be stored in `cain/src/module/`.
- The executable (`cain.exe`) is to be stored in `cain/exec/`.

For using this `Makefile` on your system, you may have to change the followings.

- The compile command `FC=f90` must be adapted to your system.
- The compiler option (`FOPT= I./module -module module` for SAD) must states that the module files (`*.mod`) are to be stored in `src/module/` (`-module module`) and, when compiling `USE` statement, the same directory must be searched for (`I./module`).
- Also, You may have to change all the file names `*.f` to `*.f90`.

The directory `cain235/exec` contains a `csh` script file `@make` although the script might be system dependent. If it does not work with minor modification, you have to write a makefile by yourself. `@make` works only when the current directory is `cain235/exec`. (You can modify it so that it works anywhere. The only problem is that **CAIN** does not know in which directory you placed him.)

When you compile all the source files, you say `@make all`, and when compiling only the files you changed, you should just say `@make`. (When `@make all` stopped due to a compilation error, `@make` will be enough next time, because `@make all` '`touche`s' all the files at the beginning.) If `@make` does not work and if you think a minor effort would be enough, please write to mailto://kaoru.yokoya@kek.jp. When you execute `@make`, a line 'sysname=....' with be shown on the monitor. Please tell me the letters on the right-hand-side to identify your system.

There used to be system-dependent subroutines for the date and the computing time. They have been replaced by the standard FORTRAN subroutines `DATE_AND_TIME` and `SYSTEM_CLOCK` since **CAIN**2.35.

4.1.3 Storage Requirements

Dynamic allocation of FORTRAN90 has been used since **CAIN**2.2 for some of the very large arrays so that you can change the size in runtime (See Sec.3.1).

The array lengths changeable in runtime are:

`MP`	Maximum number of macro particles, including photons, electrons, positrons, test particles, right-going and left-going. (Actual maximum number is 90% of `MP` because 10% is reserved for newly created particles in one time step.) 192×`MP` = 19.2MB (default: `MP`=100000)
`MVPH`	Maximum number of virtual photons in a time step in an s-slice. 80×`MVPH` = 0.8MB (default: `MVPH`=10000)
`MMAG`	Maximum number of magnets of different types.
`MBEAMLINE`	Maximum number of beamlines.
`MBBXY`	Maximum number of bins in each of x and y for beam-beam force calculation. 88×`MBBXY`^{2} = 1.37MB. (default: `MBBXY`=128)
`MLUMMESH`	Maximum number of bins in each of x and y for for luminosity calculation. 152×`MLUMMESH`^{2} = 2.4MB. (default: `MLUMMESH`=128)

Other large arrays are given by parameter statements. Major ones are the following. The given numbers are those in the present version. You can change them and re-compile all the files.

`MWLUM`	in `include/lumcom.h`. Store differential luminosity. 8×`MWLUM` = 1.6MB (`MWLUM`=200000).
`MTSTP`	in `include/tstpcm.h`. Store the history of test particles. `MTSTP` is the maximum number of the number of time steps times the number of test particles. 100×`MTSTP` = 0.5MB (`MTSTP`=5000).

The sum is about 30MB (with defaults). The size of the load module is about 0.8MB.

4.1.4 Run

All the input data have to be written in the directory `cain235/in` with file names having the extension '`.i`'. The file set sent to you may contain some example data. The directory `cain235/exec` contains a `csh` script file `@go` for execution. It works only when the current directory is `cain235/exec` like `@make`. Note that `@make` is always called from `@go` so that you do not need `@make`.

When you want to run **CAIN** with the input data `example.i`, for example, you would say `@go example` (without '`.i`'). If you use the same input file as in the previous run, `@go` suffices. TopDrawer output will be written on `cain235/out/example.tdr`, `OutFile` on `cain235/out/example` and `OutFile2` on `cain235/out/example.out2`.

If you want a submit job, please write an approproate shell script by yourself.

4.2 Windows Version

You can get the binary module of **CAIN** for Windows from the ftp site
`ftp://lcdev.kek.jp/pub/Yokoya/CainW.zip`
It is confirmed to work on Windows 2000 and XP.

4.2.1 Installation

Expanding `CainW.zip` somewhere in your hard drive to get a directory `CainW`. (Must be in hard drive since a file `cain.ini` will be created in the same directory.) No other installation process is needed.

The directory will contain

`CainW.exe`	The load module
`readme.txt`	A short memo for installation.
`example`	A directory containing an example input data.

4.2.2 Directory Structure

```
CainW ┬─src        ┬─many files   most of the fortran source files.
      │            ├─module       source files containing MODULE.
      │            ├─windows      files for Windows
      │            ├─unix         files for unix (dummies of windows subroutines).
      │            │              (These files must not be compiled.)
      │            └─include      all the files to be INCLUDEd.
      ├─Release                   contains the load module CainW.exe of release version
      ├─Debug                     contains the load module CainW.exe of debug version
      ├─resource                  contains the files to create icons.
      ├─in                        contains example data.
      ├─doc                       documents (readme.txt)
      └─other files               files of project settings. The main project file is CainW.dsw.
```

4.2.3 Run

To run **CAIN** for Windows

- From the DOS prompt, say `CainW.exe input_file_name`. (including the extension. `CainW.exe` must be in valid path.)
- Or, more conveniently, drag-and-drop the input file icon on to the icon of `CainW.exe`. (Clicking the icon of `CainW.exe` won't work.)

4.2.4 Difference of usage from UNIX version

- You must not change the variables `MsgFile`, `OutFile`, `OutFile2`, `TDFile` from the default values 6,12,12,8. `MsgFile` will appear on the console, `OutFile`=`OutFile2` and `TDFile` will be created as `*.dat` and `*.tdr` in the directory where the input file is located where `*` is the input file name without extension. If you want outputs to other files, you have to explicitly open files by using **CAIN** commands (`FILE OPEN` commad or `FILE` operand of various commands).
  Windows9x stops when an undefined file unit number such as WRITE(x,...) is encountered whereas UNIX usually automatically creates a file named `fort.x` (KEK DEC station) or `ftn00x` (KEK HP station). Due to this fact, changing the variables above will cause a file open error. [3]
- One problem of the compiler on Windows (Visual Fortran) is that the number of stored lines in the console window is very limited. When the output to `MsgFile` exceeds some hundred lines, the early are will be lost. If you want the destination of `MsgFile` to be a file, please insert the following line in the file `Cain.ini` (in the same directory where `Cain.exe` is located)

```
                    MSGDEST=FILE
```

[3]It is, of course possible to open files explicitly in FORTRAN instead of defining them by shell environment variables as is done now in the `@go` command. However, this would cause a change of usage in the UNIX side, which I do not want. I want the source files to be identical except for the two files above.

(The default is `MSGDEST=CONSOLE`.) Then the messages will go to `*.txt` in the same directory as `*.dat`. The same message will also go to the console [4] with some delay. (The messages to the file are copied at every encounter of **CAIN** command.)

- The current directory is the directory where the input file is located rather than `cain/exec`. Be carefull with the file name. You do not need directories such as `cain/in/` and `cain/out/` unless you want.

- Since only the binary module is distributed, you cannot change the size of arrays. However, since **CAIN**2.2, you can dynamically allocate the arrays related to the maximum number of macro-particles so that there should be no serious problem. If you still want differefent sizes for other arrays, please email to `mailto:kaoru.yokoya@kek.jp`.

4.2.5 TopDrawer

Once you run **CAIN** on PC, you may want to view the TopDrawer output on the same platform. An incomplete TopDrawer for Windows is available. If you want it in spite of full of bugs and danger, go to the ftp site
`ftp://lcdev.kek.jp/pub/Yokoya/TopDrawW.zip`

[4] This is not true with **CAIN**2.32.

Chapter 5

Physics and Numerical Methods

5.1 Coordinate

One of the basic assumptions of **CAIN** is that the main part (i.e., the part which contributes to the beam field dominantly) of the high energy beams consists of either (almost) right-going or left-going particles. The longitudinal coordinate s is the right-going direction. (The reason s is used instead of z is only historical since **ABEL**.) The x and y axes are perpendicular to s and (x, y, s) forms a right-handed orthonormal frame. The time coordinate t is always multiplied by the velocity of light.

In contrast to **ABEL**, **CAIN** does not use the longitudinal coordinate (z_1, z_2) attached to the beams.

5.2 Particle Variables

5.2.1 Arrays for Particles

All the particles (photons, electrons, positrons) carry the following variables.

`TXYS`(i)	$(i = 0, 1, 2, 3)$ Particle coordinates in meter. Note that, in contrast to **ABEL**, the time and the s-coordinates are also defined for each particle. During tracking by `PUSH-ENDPUSH` command all the particles have basically the same time coordinate (an exception is the particles just created), whereas in some cases (e.g., after defined by `BEAM` command, after `DRIFT S=`s_1 command, etc.) they have different t but same s. Also note that, in contrast to **ABEL**, s-coordinate does not simply change as $s_0 \pm ct$ but changes according to the instantaneous longitudinal velocity so that longitudinal mixing may occur for low energy or large angle particles.
`EP`(i)	$(i = 0, 1, 2, 3)$ Energy-momentum in units of (eV,eV/c).
`SPIN`(i)	$(i = 1, 2, 3)$ The polarization component (S_x, S_y, S_s) for electrons/positrons, and the Stokes parameter (ξ_1, ξ_2, ξ_3) for photons. (S_x, S_y, S_s) is defined, as usual, in particle's rest frame. Therefore, it aquires

the Thomas precession under Lorentz transformation by `LORENTZ` command.

For defining the Stokes parameter, one needs a set of orthonormal basis vectors $(\boldsymbol{e}^{(1)}, \boldsymbol{e}^{(2)}, \boldsymbol{e}^{(3)})$ with the third vector $\boldsymbol{e}^{(3)}$ parallel to the momentum. In **CAIN**, the first vector $\boldsymbol{e}^{(1)}$ is taken to be the unit vector along $\boldsymbol{e}_x - \boldsymbol{e}^{(3)}(\boldsymbol{e}_x \cdot \boldsymbol{e}^{(3)})$ and $\boldsymbol{e}^{(2)} = \boldsymbol{e}^{(3)} \times \boldsymbol{e}^{(1)}$. This is ill-defined when the momentum is exactly parallel to the x-axis but this possibility is simply ignored. Except for large angle photons, $(\boldsymbol{e}^{(1)}, \boldsymbol{e}^{(2)}, \boldsymbol{e}^{(3)})$ is almost equal to $(\boldsymbol{e}_x, \boldsymbol{e}_y, \boldsymbol{e}_z)$ for right-going photons, and $(\boldsymbol{e}_x, -\boldsymbol{e}_y, -\boldsymbol{e}_z)$ for left-going photons.

See the next subsection for more detail on the polarization.

`GEN` Generation. When a particle is generated by `BEAM` command by Twiss parameters, etc, `GEN`=1. Created particles such as beamstrahlung photons have `GEN` larger by one than that of the parent particle. (This is also true for the 'spent' parents.) `GEN` of the secondary particles due to particle-particle interaction (such as incoherent pairs) is the sum of `GEN`s of the parents. In this case `GEN`s of the parents do not change.

`WGT` Weight. Number of real particles represented by the macro-particle.

`NAME` 4-byte character string. Normally blanks. The test particles have `Tnnn` where `nnn` is a three-digit number. `NAME` of the particles created by incoherent (particle-particle) interactions starts with '`I`'. For example, '`IBW `', '`IBH `', '`ILL `' for the pairs created by incoherent Breit-Wheeler, Bethe-Heitler, Landau-Lifshitz processes, respectively.

5.2.2 Description of Polarization

Convention for electron/positron polarization

In most applications, one is interested in the helicity states. Therefore, one possible way of expressing the electron/positron spin is to store the information whether each macro-particle is in the helicity $h = +1$ state or -1 state. The unpolarized state is represented by an equal number of macro-particles with $h = +1$ and -1. The spin may flip at the interactions such as laser-Compton scattering and beamstrahlung.

However, this simple way cannot be applied to our case because, for example, a pure transverse polarization may become longitudinal during the precession in a magnetic field (beam-beam field or external field). In order to include such classical precession effects, the phase relation between the up and down components of the spinor is important.

This problem can be solved by using the density matrix. Let us express an electron(positron) state by a two-component spinor φ. The 2×2 density matrix $\rho^{(e)}$ is defined as

$$\rho^{(e)}_{ij} = \left\langle \varphi_i \varphi_j^\dagger \right\rangle, \qquad (i, j = 1, 2) \tag{5.1}$$

where $\dagger$ denotes the Hermitian conjugate and $\langle\ \rangle$ is the average over a particle ensemble. Since $\rho^{(e)}$ is Hermitian and its trace is unity by normalization, $\rho^{(e)}$ can be written as

$$\rho^{(e)} = \tfrac{1}{2}(1 + \boldsymbol{\zeta} \cdot \boldsymbol{\sigma}), \qquad \boldsymbol{\zeta} = \mathrm{Trace}(\rho^{(e)} \boldsymbol{\sigma}) = \left\langle \varphi \boldsymbol{\sigma} \varphi^\dagger \right\rangle \tag{5.2}$$

where $\boldsymbol{\sigma}$ is the Pauli matrices. The 3-vector $\boldsymbol{\zeta}$ is called polarization vector.

In the case of pure states, φ can be represented by a superposition of spin up(down) states $\varphi_\pm$:

$$\varphi = c_+\varphi_+ + c_-\varphi_-, \qquad |c_+|^2 + |c_-|^2 = 1. \tag{5.3}$$

With the standard representation of the Pauli matrices

$$\sigma_1 = \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix}, \quad \sigma_2 = \begin{pmatrix} 0 & -i \\ i & 0 \end{pmatrix}, \quad \sigma_3 = \begin{pmatrix} 1 & 0 \\ 0 & -1 \end{pmatrix}, \tag{5.4}$$

$\boldsymbol{\zeta}$ can be written as

$$\zeta_1 = 2\Re(c_+^* c_-), \quad \zeta_2 = 2\Im(c_+^* c_-), \quad \zeta_3 = |c_+|^2 - |c_-|^2, \tag{5.5}$$

and its length is unity: $|\boldsymbol{\zeta}| = 1$. **CAIN** allows $|\boldsymbol{\zeta}| \leq 1$ so that each macro-particle is in a mixed state, representing an ensemble of particles having almost the same energy-momentum and space-time coordinate.

If one observes the particle spin with the quantization axis $\boldsymbol{e}$ ($|\boldsymbol{e}| = 1$), the probability to be found in the spin $\pm\boldsymbol{e}$ state is given by $(1 \pm \boldsymbol{\zeta}\cdot\boldsymbol{e})/2$.

The polarization vector $\boldsymbol{\zeta}$ obeys the Thomas-BMT equation (5.34) in the absense of quantum phenomena.

Convention for photon polarization

A similar way is used for photon polarization, too. The polarization vector (3-vector) $\boldsymbol{\epsilon}$ (normalized as $|\boldsymbol{\epsilon}| = 1$) is orthogonal to the photon momentum $\boldsymbol{k}$. It can be represented by the components along two unit vectors $\boldsymbol{e}^{(1)}$ and $\boldsymbol{e}^{(2)}$ perpendicular to $\boldsymbol{k}$. The three vectors ($\boldsymbol{e}^{(1)}$, $\boldsymbol{e}^{(2)}$, $\boldsymbol{k}/|\boldsymbol{k}|$) form a right-handed orthonormal basis. The density matrix is defined as

$$\rho_{ij}^{(\gamma)} = \langle(\boldsymbol{\epsilon}\cdot\boldsymbol{e}_i)(\boldsymbol{\epsilon}^*\cdot\boldsymbol{e}_j)\rangle\,. \tag{5.6}$$

This is Hermitian with unit trace as in the case of electron density matrix so that it can be written as

$$\rho^{(\gamma)} = \tfrac{1}{2}(1 + \boldsymbol{\xi}\cdot\boldsymbol{\sigma}), \qquad \boldsymbol{\xi} = \mathrm{Trace}(\rho^{(\gamma)}\boldsymbol{\sigma}). \tag{5.7}$$

The 3-vector $\boldsymbol{\xi}$ is called the Stokes parameter. In the standard representation of the Pauli matrices, the three components of $\boldsymbol{\xi}$ have the meaning

ξ_1	Linear polarization along the direction $(\boldsymbol{e}^{(1)} + \boldsymbol{e}^{(2)})/\sqrt{2}$ ($\xi_1 > 0$) or $(\boldsymbol{e}^{(1)} - \boldsymbol{e}^{(2)})/\sqrt{2}$ ($\xi_1 < 0$)
ξ_2	Circular polarization
ξ_3	Linear polarization along the direction $\boldsymbol{e}^{(1)}$ ($\xi_3 > 0$) or $\boldsymbol{e}^{(2)}$ ($\xi_3 < 0$).

The linear polarization can also be written as $\xi_3 = \xi_L \cos 2\phi_L$ and $\xi_1 = \xi_L \sin 2\phi_L$ ($\xi_L \geq 0$) where ξ_L is the magnitude of linear polarization and ϕ_L (modulo π) is the angle of the polarization plane measured from the $\boldsymbol{e}^{(1)}$-axis counterclockwise.

Completely polarized states have $|\boldsymbol{\xi}| = 1$. A single photon is always in a completely polarized state. Mixed states may have $|\boldsymbol{\xi}| < 1$.

In contrast to the case of electron/positron the polarization of a photon with a given momentum cannot be defined by the three numbers ξ_i: one has to define the $\boldsymbol{e}^{(1)}$-axis. The most general way is that every macro-photon carries its own $\boldsymbol{e}^{(1)}$-axis but this is too much redundant. **CAIN** adopts the convention that $\boldsymbol{e}^{(1)}$ is parallel to $\boldsymbol{e}_x - (\boldsymbol{e}_x{\cdot}\boldsymbol{k})\boldsymbol{k}/\,|\boldsymbol{k}|^2$ where $\boldsymbol{k}$ is the photon momentum. This is ill-defined when $\boldsymbol{k}$ is parallel to $\boldsymbol{e}_x$ but it will not cause a serious problem. (For lasers $\boldsymbol{e}^{(1)}$-axis must be specified explicitly.)

Polarization-related processes

In any process involving polarizations, the transition rate (or crosssection) is given by multiplying the density matrices and by taking the trace. Therefore, the expressions for the rates are bilinear forms for each polarization vector, initial/final electron/positron or photon. The final polarization needs some comments. The transition rate is written in general as

$$W = \frac{1}{2}\int d\Gamma (w + \boldsymbol{g}{\cdot}\overline{\boldsymbol{\zeta}}) \tag{5.8}$$

where Γ represents the final energy-momentum variables and w and $\boldsymbol{g}$ are functions of Γ. The vector $\overline{\boldsymbol{\zeta}}$ itself is not the final polarization. Its direction is defined by the setup of the detectors. What the term $\boldsymbol{g}{\cdot}\overline{\boldsymbol{\zeta}}$ means is that, if one observes the spin direction $\boldsymbol{e}$ ($|\boldsymbol{e}| = 1$), the probability to be found in the state $\pm\boldsymbol{e}$ is given by $\frac{1}{2}\int d\Gamma(w \pm \boldsymbol{g}{\cdot}\boldsymbol{e})$.

The final energy-momentum distribution is determined by $w(\Gamma)$. For given Γ, the final polarization vector is (see [3], page 254)

$$\boldsymbol{\zeta} = \boldsymbol{g}(\Gamma)/w(\Gamma). \tag{5.9}$$

Now, consider a process involving initial and final electrons, summing over other possible particles. The transition rate is written as

$$dW = \frac{1}{2}\int d\Gamma (w + \boldsymbol{f}{\cdot}\boldsymbol{\zeta}_i + \boldsymbol{g}{\cdot}\overline{\boldsymbol{\zeta}}_f + \overline{\boldsymbol{\zeta}}_f^T H \boldsymbol{\zeta}_i) \tag{5.10}$$

where the subscripts i and f denote initial and final variables, ^{T} represents transpose, and H is a 3×3 matrix. For given $\boldsymbol{\zeta}_i$, the final energy-momentum distribution is determined by $w + \boldsymbol{f}{\cdot}\boldsymbol{\zeta}_i$. In a Monte Carlo algorithm, Γ is decided by using random numbers according to $w + \boldsymbol{f}{\cdot}\boldsymbol{\zeta}_i$. Once Γ is decided, the final polarization is definitely (without using random numbers) given by

$$\boldsymbol{\zeta}_{f,\mathrm{trans.}} = \frac{\boldsymbol{g}(\Gamma) + H(\Gamma)\boldsymbol{\zeta}_i}{w(\Gamma) + \boldsymbol{f}(\Gamma){\cdot}\boldsymbol{\zeta}_i}. \tag{5.11}$$

This expression does not satisfy $|\boldsymbol{\zeta}| = 1$. If one does not allow a macro-particle in a mixed state, one has to choose a pure state by using random numbers.

The macro-particles which did not make transition must carefully be treated. One might say their final polarization is equal to $\boldsymbol{\zeta}_i$ but this is not correct because of the selection effect due to the term $\boldsymbol{f}\cdot\boldsymbol{\zeta}_i$.

The probability that a transion does not occur in a time interval Δt is $1-(\underline{w}+\underline{\boldsymbol{f}}\cdot\boldsymbol{\zeta}_i)\Delta t$, where the underlines indicates quantities integrated over the whole kinetic range of $\mathit{\Gamma}$. Consider an ensemble (one macro-particle) of N (real) particles having the polarization vector $\boldsymbol{\zeta}_{i,\alpha}$ ($\alpha = 1, 2, \ldots, N$). Each of these is a unit vector $|\boldsymbol{\zeta}_{i,\alpha}| = 1$ and the average over the ensemble is $\boldsymbol{\zeta}_i = \langle\boldsymbol{\zeta}_{i,\alpha}\rangle$.

Let us arbitrarily take the quatization axis $\boldsymbol{e}$. The probability in the state $\pm\boldsymbol{e}$ is $(1 \pm \boldsymbol{e}\cdot\boldsymbol{\zeta}_{i,\alpha})/2$ and the non-transition probability is $1 - (\underline{w} \pm \underline{\boldsymbol{f}}\cdot\boldsymbol{e})\Delta t$. Therefore, the sum of the final polarization along $\boldsymbol{e}$ over the ensemble is

$$\sum_\alpha\sum_\pm \pm\frac{1 \pm \boldsymbol{e}\cdot\boldsymbol{\zeta}_{i,\alpha}}{2}[1 - (\underline{w} \pm \underline{\boldsymbol{f}}\cdot\boldsymbol{e})\Delta t] = \sum_\alpha[\boldsymbol{e}\cdot\boldsymbol{\zeta}_{i,\alpha}(1 - \underline{w}\Delta t) - \underline{\boldsymbol{f}}\cdot\boldsymbol{e}\Delta t] = N\boldsymbol{e}\cdot[\boldsymbol{\zeta}_i(1 - \underline{w}\Delta t) - \underline{\boldsymbol{f}}\Delta t].$$

The axis $\boldsymbol{e}$ is arbitrary. Therefore, the sum of the final polarization vector is given by the above expression with $\boldsymbol{e}$ taken away. The total number of particles without transition is $N[1 - (\underline{w} + \underline{\boldsymbol{f}}\cdot\boldsymbol{\zeta}_i)\Delta t]$. Thus, the final polarization vector is

$$\boldsymbol{\zeta}_{f,\text{no trans.}} = \frac{\boldsymbol{\zeta}_i(1 - \underline{w}\Delta t) - \underline{\boldsymbol{f}}\Delta t}{1 - (\underline{w} + \underline{\boldsymbol{f}}\cdot\boldsymbol{\zeta}_i)\Delta t} \tag{5.12}$$

The average final polarization over the whole ensemble, with and without transition, is then given by

$$\begin{aligned}\boldsymbol{\zeta}_f &= [1 - (\underline{w} + \underline{\boldsymbol{f}}\cdot\boldsymbol{\zeta}_i)\Delta t]\,\boldsymbol{\zeta}_{f,\text{notrans.}} + \int d\mathit{\Gamma}[w(\mathit{\Gamma}) + \boldsymbol{f}(\mathit{\Gamma})\cdot\boldsymbol{\zeta}_i]\Delta t\,\boldsymbol{\zeta}_{f,\text{trans.}} \\ &= \boldsymbol{\zeta}_i + [\underline{\boldsymbol{g}} - \underline{\boldsymbol{f}} - \underline{w}\boldsymbol{\zeta}_i + \underline{H}\boldsymbol{\zeta}_i]\Delta t. \end{aligned} \tag{5.13}$$

If one can ignore the change of energy-momentum during the transition, the evolution of the polarization is described by the differential equation

$$\frac{d\boldsymbol{\zeta}}{dt} = \underline{\boldsymbol{g}} - \underline{\boldsymbol{f}} - \underline{w}\boldsymbol{\zeta} + \underline{H}\boldsymbol{\zeta}. \tag{5.14}$$

<u>Polarization effects included in **CAIN**</u>

The present version of **CAIN** does not include all the polarization effects. The following table shows what effects are included. In any case, the correlation of polarization between final particles is not taken into account.

		initial $e^\pm$	laser	final $e^\pm$	final γ
Beamstrahlung	$e^\pm \to e^\pm + \gamma$	LT	–	LT	LT
Linear laser-Compton	$e^\pm$ + laser $\to e^\pm + \gamma$	LT	LT	LT	LT
Nonlinear laser-Compton	$e^\pm + n\cdot$laser $\to e^\pm + \gamma$	L	L*	L	L
	or	N	T*	N	T
		initial γ	laser	final $e^\pm$	
Coherent pair	$\gamma \to e^+ + e^-$	LT	–	LT	
Linear laser-Breit-Wheeler	γ + laser $\to e^+ + e^-$	LT	LT	LT	
Nonlinear laser-Breit-Wheeler	$\gamma + n\cdot$laser $\to e^+ + e^-$	L	L*	L	
		initial	final pair		
Incoherent Breit-Wheeler	$\gamma + \gamma \to e^+ + e^-$	L	N		
Incoherent Bethe-Heitler	$\gamma + e \to e + e^+ + e^-$	N	N		
Incoherent Landau-Lifshitz	$e + e \to e + e + e^+ + e^-$	N	N		
		initial	final		
Bremsstrahlung	$e + e \to e + e + \gamma$	N	N		

L	Longitudinal spin of electron/positron (or circular polarization of photon).
T	Transverse spin of electron/positron (or linear polarization of photon).
*	±100% polarization only.
N	Not computed. (No change for existing particles, zero for created particles)
–	Irrelevant.

5.3 Beam Parameters

The BEAM command makes it possible to define a beam in terms of the conventional Twiss parameters. A beam is defined by many parameters described in Sec.3.5. Here, we will give formulas how to generate a beam using these parameters. An important point is that the beam is defined on a plane s=constant rather than t=constant. Thus, the longitudinal structure of the beam appears as the time structure. Note that t is larger at the bunch tail.

The parameters are

(t_0, x_0, y_0, s_0)	Reference point of the Twiss parameters. (m)
E_0	Reference energy. (eV)
$\beta_{x,y}, \alpha_{x,y}, \eta_{x,y}, \eta'_{x,y}$	Twiss parameters.
$\epsilon_{x,y}$	Geometric emittance. (rad·m)
σ_t	R.m.s. bunch length. (m)
σ_ε	R.m.s. relative energy spread.
$n_x, n_y, n_t, n_\varepsilon$	Gaussian tail cut off.
θ_x, θ_y	Orbit slope. (rad)
ψ_x, ψ_y	Crab angle. (rad)
ϕ_{xy}	Beam roll in the x-y plane. (rad)

$d\varepsilon/dt$	Coherent energy slope (1/m).
$d\alpha_{xy}/d\varepsilon$	Energy dependence of the focal point. (Parameter for the travelling focus.)

First, generate particle variables in usual accelerator coordinate:

$$t_1 = \sigma_t r_1 \tag{5.15}$$
$$\varepsilon_1 = \sigma_\varepsilon r_2 + (d\varepsilon/dt)t_1 \tag{5.16}$$
$$x_1 = \sqrt{2\epsilon_x u_1 \beta_{x1}}\cos\varphi_1 + \eta_x\varepsilon_1 \tag{5.17}$$
$$x_1' = \sqrt{2\epsilon_x u_1/\beta_{x1}}(-\alpha_{x1}\cos\varphi_1 - \sin\varphi_1) + \eta_x'\varepsilon_1 \tag{5.18}$$
$$y_1 = \sqrt{2\epsilon_y u_2 \beta_{y1}}\cos\varphi_2 + \eta_y\varepsilon_1 \tag{5.19}$$
$$y_1' = \sqrt{2\epsilon_y u_2/\beta_{y1}}(-\alpha_{y1}\cos\varphi_2 - \sin\varphi_2) + \eta_y'\varepsilon_1 \tag{5.20}$$
$$\alpha_{xy1} = \alpha_{xy} + \varepsilon_1(d\alpha_{xy}/d\varepsilon), \quad \beta_{xy1} = \beta_{xy}\left[1 + (\varepsilon_1 d\alpha_{xy}/d\varepsilon)^2\right] \tag{5.21}$$

where r_1 (r_2) is a Gaussian random number of zero mean and unit standard deviation cut at n_t (n_ε), u_j (j=1,2) is a random number of exponential distribution ($\propto e^{-u}$) cut at $u = n_x^2/2, n_y^2/2$ and φ_j a uniform random number in $(0, 2\pi)$.

These variables are then transformed to

$$t = t_0 + t_1 \tag{5.22}$$
$$\begin{pmatrix} x \\ y \end{pmatrix} = \begin{pmatrix} x_0 \\ y_0 \end{pmatrix} + \begin{pmatrix} \psi_x \\ \psi_y \end{pmatrix} t_1 + \begin{pmatrix} \cos\phi_{xy} & -\sin\phi_{xy} \\ \sin\phi_{xy} & \cos\phi_{xy} \end{pmatrix} \begin{pmatrix} x_1 \\ y_1 \end{pmatrix} \tag{5.23}$$
$$\begin{pmatrix} x' \\ y' \end{pmatrix} = \begin{pmatrix} \theta_x \\ \theta_y \end{pmatrix} + \begin{pmatrix} \cos\phi_{xy} & -\sin\phi_{xy} \\ \sin\phi_{xy} & \cos\phi_{xy} \end{pmatrix} \begin{pmatrix} x_1' \\ y_1' \end{pmatrix} \tag{5.24}$$
$$s = s_0 \tag{5.25}$$

Finally, the energy-momentum is given by

$$E = E_0 + E_0\varepsilon_1 \tag{5.26}$$
$$p_s = \pm\sqrt{\frac{E^2 - m^2}{1 + x'^2 + y'^2}} \tag{5.27}$$
$$p_x = |p_s|\, x' \tag{5.28}$$
$$p_y = |p_s|\, y' \tag{5.29}$$

where $\pm$ is $+$ for right-going beam and $-$ for left-going (note that right/left appears only here) and m is the relevant particle mass in units of eV/c^2.

5.4 Solving Equation of Motion

Under the `PUSH` command, the equation of particle motion is solved step by step with the `Time` as the independent variable. The time step size is determined automatically for each particle. Smaller step size is used for low energy particles. On the other hand, an exact solution is used in the case of `DRIFT EXTERNAL` command which uses either the time or s as the independent variable.

5.4.1 Equation of motion under `DRIFT EXTERNAL` command

The present version of **CAIN** accepts a constant external field only. The covariant form of the equation of motion

$$\frac{dx^\mu}{d\tau} = \frac{1}{m}p^\mu, \qquad \frac{dp^\mu}{d\tau} = \frac{e}{m}F^\mu{}_\nu p^\nu. \tag{5.30}$$

where τ is the proper time and F^μ_ν the electromagnetic field tensor, can be solved exactly when the field is constant. The eigenvalues of the matrix $f^\mu{}_\nu \equiv eF^\mu{}_\nu/m$ is given by $\pm\omega_1$ and $\pm\omega_2$, where

$$\omega_1 = \sqrt{\tfrac{1}{2}(\sqrt{a^2+4b^2}+a)}, \qquad \omega_2 = i\sqrt{\tfrac{1}{2}(\sqrt{a^2+4b^2}-a)} \tag{5.31}$$

with

$$a = \frac{e^2}{m^2}(\boldsymbol{E}^2 - \boldsymbol{B}^2), \quad b = \frac{e^2}{m^2}(\boldsymbol{E}\cdot\boldsymbol{B}). \tag{5.32}$$

Then, the solution is

$$p^\mu(\tau) = \sum_{\omega=\omega_1,\omega_2} \frac{\pm 1}{\sqrt{a^2+4b^2}} \left[(\omega^2 + \tilde{f}^\mu{}_\alpha \tilde{f}^\alpha{}_\nu)\cosh\omega\tau + (\omega^2 f^\mu{}_\nu + b\tilde{f}^\mu{}_\nu)\frac{\sinh\omega\tau}{\omega}\right] p^\nu(0), \tag{5.33}$$

where the upper (lower) sign applies to ω_1 (ω_2) and $\tilde{f}^{\mu\nu} \equiv \frac{1}{2}\epsilon^{\mu\nu\alpha\beta}f_{\alpha\beta}$ with $\epsilon^{\mu\nu\alpha\beta}$ being the antisymmetric tensor of rank 4.

The classical spin motion of electrons is given by the Thomas-BMT equation

$$\frac{d\boldsymbol{S}}{dt} = -\frac{e}{m\gamma}\left[(\gamma a+1)\boldsymbol{B}_T + (a+1)\boldsymbol{B}_L - \gamma(a+\frac{1}{\gamma+1})\beta\boldsymbol{e}_v\times\frac{\boldsymbol{E}}{c}\right]\times\boldsymbol{S}, \tag{5.34}$$

where a is the coefficient of anomalous magnetic moment and

$$\boldsymbol{B}_L = \boldsymbol{e}_v(\boldsymbol{B}\cdot\boldsymbol{e}_v), \qquad \boldsymbol{B}_T = \boldsymbol{B} - \boldsymbol{B}_L = \boldsymbol{e}_v\times(\boldsymbol{B}\times\boldsymbol{e}_v). \tag{5.35}$$

When the field is very strong, a is different from the well-known value $\alpha/2\pi + O(\alpha^2)$ but is a function of the field strength characterized the parameter Υ.

$$\Upsilon = \frac{e}{m^3}\sqrt{-(F^{\mu\nu}p_\nu)^2} = \frac{e}{m^3}\sqrt{(p^0\boldsymbol{E}_T + \boldsymbol{p}\times\boldsymbol{B})^2 + \boldsymbol{E}_L^2}. \tag{5.36}$$

According to V. N. Baier,

$$\frac{a(\Upsilon)}{a(0)} = \frac{2}{\Upsilon}\int_0^\infty \frac{xdx}{(1+x)^3}\int_0^\infty \sin\left[\frac{x}{\Upsilon}\left(t+\frac{t^3}{3}\right)\right]dt \tag{5.37}$$

The function $a(\Upsilon)/a(0)$ is shown in Fig.5.1. Simple polynomial approximations are used in **CAIN**.

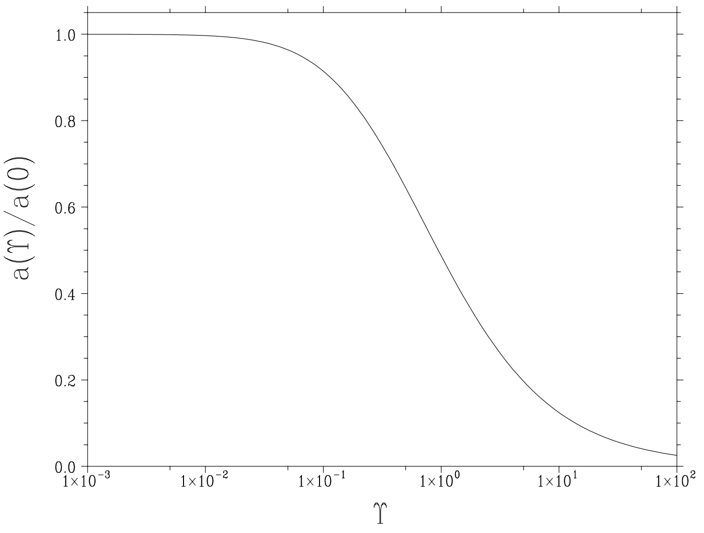


Figure 5.1: Field dependence of the anomalous magnetic moment of electron

5.4.2 Equation of motion under PUSH command

Solving the equation of motion in PUSH command is much more complicated because of the possible presense of the beam field. The equation of motion is in general written in the form

$$\frac{d\boldsymbol{r}}{dt} = \boldsymbol{v}(\boldsymbol{p}) = \frac{\boldsymbol{p}}{\sqrt{m^2+\boldsymbol{p}^2}} \tag{5.38}$$

$$\frac{d\boldsymbol{p}}{dt} = \boldsymbol{F}(\boldsymbol{r},\boldsymbol{p}) \tag{5.39}$$

The force $\boldsymbol{F}$ includes the beam field and the external field. The $\boldsymbol{p}$ dependence of $\boldsymbol{F}$ comes from $\boldsymbol{v}\times\boldsymbol{B}$ although very weak in the case of the beam field.

Given the initial variables $(\boldsymbol{r}_0,\boldsymbol{p}_0)$, a simple approximation after the time interval Δt is

$$\begin{aligned}
\boldsymbol{F}_0 &= \boldsymbol{F}(\boldsymbol{r}_0,\boldsymbol{p}_0)\\
\boldsymbol{p}_1 &= \boldsymbol{p}_0+\boldsymbol{F}_0\Delta t\\
\boldsymbol{r}_1 &= \boldsymbol{r}_0+\tfrac{1}{2}[\boldsymbol{v}(\boldsymbol{p}_0)+\boldsymbol{v}(\boldsymbol{p}_1)]\Delta t\\
\boldsymbol{F}_1 &= \boldsymbol{F}(\boldsymbol{r}_1,\boldsymbol{p}_1)\\
\boldsymbol{p} &= \boldsymbol{p}_0+\tfrac{1}{2}(\boldsymbol{F}_0+\boldsymbol{F}_1)\Delta t\\
\boldsymbol{r} &= \boldsymbol{r}_0+\tfrac{1}{2}[\boldsymbol{v}(\boldsymbol{p}_0)+\boldsymbol{v}(\boldsymbol{p})]\Delta t.
\end{aligned}$$

The error of $\boldsymbol{r}$ by these formulas is estimated by

$$\delta\boldsymbol{r} = \frac{1}{4}\frac{\boldsymbol{F}_1-\boldsymbol{F}_0}{\sqrt{m^2+\boldsymbol{p}_1^2}}(\Delta t)^2.$$

If this is not small enough, divide the interval Δt by an integer n_d. Note that $\delta \boldsymbol{r} \propto (\Delta t)^3$ because $\boldsymbol{F}_1 - \boldsymbol{F}_0$ is proportional to Δt. The total error, after multiplied by the number of intervals n_d, is proportional to $1/n_d^2$.

However, the above prescription is not really enough when there are extremely low energy particles (e.g., those from incoherent pair creation). It often happens that n_d so determined bocomes over several hundreds. In such a case the above error estimation may not be accurate at all.

When n_d is too large, **CAIN** tries the fourth-order Runge-Kutta integration. Starting from the whole interval Δt, it is divided by 2 at each step until the difference becomes small enough. This method is a little better than the simple formulas above but is still time consuming. So, the users should be aware that incoherent pair creation is expensive.

5.5 Beamline

A beamline consists of magnets, RF cavities, drift spaces, etc. Here, we shall call them 'beamline elements' or simply 'elements', and sometimes 'magnet'. (RF cavities are not ready yet.) In CAIN only single particle dynamics in given fields is taken into account.

5.5.1 Beamline Coordinate

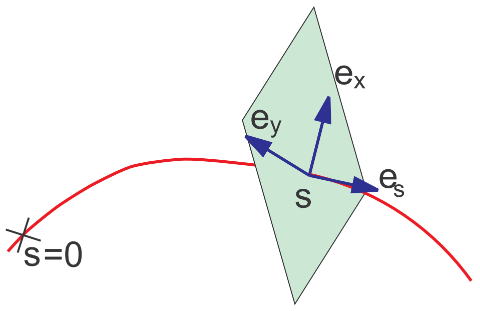
A beamline has a reference orbit which is defined by the element length (`LENGTH` in `MAGNET` command), orbit bending angle (`ANGLE`), and axial rotation angle (`ROTATION`). So far the reference orbit is defined only geometrically (because time-varying fields are not included yet).

5.5.2 Beamline coordinate

A beamline has is own coordinate system (t, x, y, s) which is in general a curvilinear coordinate, different from the Cartesian coordinate of CAIN. (At present the time coordinate is the same.)

Let us denote the reference orbit by $\boldsymbol{r}_0(s)$ where s is the length measured along the reference orbit from the beamline entrance. The unit vector along the orbit is

$$\frac{d\boldsymbol{r}_0}{ds} = \boldsymbol{e}_s(s) \tag{5.40}$$



One must define two vectors $\boldsymbol{e}_x(s)$ and $\boldsymbol{e}_y(s)$ perpendicular to $\boldsymbol{e}_s(s)$. Since $|\boldsymbol{e}_s(s)| = 1$, its change (orbit bending) is perpendicular to $\boldsymbol{e}_s(s)$. Therefore, one can write as

$$\frac{d\boldsymbol{e}_s}{ds} = \boldsymbol{\Omega}(s) \times \boldsymbol{e}_s, \qquad \boldsymbol{\Omega}(s) \cdot \boldsymbol{e}_s = 0 \tag{5.41}$$

The orientation of $\boldsymbol{e}_x(s)$ and $\boldsymbol{e}_y(s)$ in the plane perpendicular to $\boldsymbol{e}_s$ is defined such that the similar equation hold for all three axes:

$$\frac{d\boldsymbol{e}_j}{ds} = \boldsymbol{\Omega}(s)\times\boldsymbol{e}_j \qquad (j = x, y, s) \tag{5.42}$$

with the same $\boldsymbol{\Omega}(s)$. One can expand $\boldsymbol{\Omega}(s)$ as

$$\boldsymbol{\Omega} = -\frac{\boldsymbol{e}_y(s)}{\rho_x(s)} + \frac{\boldsymbol{e}_x(s)}{\rho_y(s)}. \tag{5.43}$$

which defines the curvature radius (ρ_x, ρ_y). By using these definitions, one can write any point vector near the reference orbit by three numbers (x, y, s) as

$$\boldsymbol{r} = \boldsymbol{r}_0(s) + x\boldsymbol{e}_x(s) + y\boldsymbol{e}_y(s) \tag{5.44}$$

In a bending magnet specified by `LENGTH`$=l$, `ANGLE`$=\theta$, `ROTATION`$=\phi$, $\boldsymbol{\Omega}(s)$ is written as

$$\boldsymbol{\Omega}(s) = \frac{1}{\rho}(\boldsymbol{e}_x\sin\phi - \boldsymbol{e}_y\cos\phi), \qquad \rho = \frac{l}{\theta} \tag{5.45}$$

By solving eqs.(5.40) and (5.42) with eq.(5.45) one finds the transportation of the basis vectors and $\boldsymbol{r}_0(s)$ through the bending magnet as

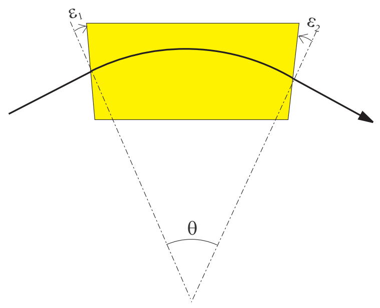
$$\begin{pmatrix}\boldsymbol{e}_x\\ \boldsymbol{e}_y\\ \boldsymbol{e}_s\end{pmatrix}_{s_0+l} = \begin{pmatrix}\cos\phi & -\sin\phi & 0\\ \sin\phi & \cos\phi & 0\\ 0 & 0 & 1\end{pmatrix}\begin{pmatrix}\cos\theta & 0 & \sin\theta\\ 0 & 1 & 0\\ -\sin\theta & 0 & \cos\theta\end{pmatrix}\begin{pmatrix}\cos\phi & \sin\phi & 0\\ -\sin\phi & \cos\phi & 0\\ 0 & 0 & 1\end{pmatrix}\begin{pmatrix}\boldsymbol{e}_x\\ \boldsymbol{e}_y\\ \boldsymbol{e}_s\end{pmatrix}_{s_0} \tag{5.46}$$

$$\boldsymbol{r}_0(s_0 + l) = \boldsymbol{r}_0(s_0) + \rho[\boldsymbol{e}_s(s_0)\sin\theta + (1-\cos\theta)(\boldsymbol{e}_x(s_0)\cos\phi + \boldsymbol{e}_y(s_0)\sin\phi)] \tag{5.47}$$

where s_0 is the entrance of the magnet.

To locate the beamline in the CAIN coordinate system, we need $\boldsymbol{r}_0$ and $(\boldsymbol{e}_x, \boldsymbol{e}_y, \boldsymbol{e}_s)$ at the entrance of the beamline. These are specified in the `TRANSPORT` command (not in `BEAMLINE` command) by the operands `TXYS`, `E3`, and `E1`. This information is not needed for the optics of the beamline.

5.5.3 Dipole Magnets



A dipole magnet is divided into three parts: the leading edge, the body (sector bend), and the trailing edge. The edges are approximated by quadrupole magnets (thin lens of inverse focal length $k_1 = -\tan\epsilon/\rho_0$, ϵ:edge angle) Since **CAIN** has to handle particles with a vast range of momentum (even from a few MeV to TeV simultaneously), an exact solution is used for the body.

Let us assume the body is defined by the horizontal bending angle θ_0 (for the reference particle with mass m_0 and momentum p_0) and the orbit length l_0 so that the curvature radius is $\rho_0 = l_0/\theta_0$. (Vertical bend is treated by coordinate rotation before and after the magnet.) Suppose that a particle enters the body at (t, x, y, s) with (E, p_x, p_y, p_s) $(p_s > 0)$. Let us define

$$\alpha \equiv \tan^{-1}(p_x/p_s), \qquad p_1 \equiv \sqrt{p_s^2 + p_x^2}, \qquad \rho \equiv \pm\rho_0 \frac{p_1}{p_0}, \qquad \omega_1 \equiv \frac{p_1}{\rho E},$$
$$f \equiv \sin(\alpha + \theta_0) - \frac{x + \rho_0}{\rho} \sin\theta_0 = 2\sin\frac{\alpha}{2}\cos(\frac{\alpha}{2} + \theta_0) + \frac{\rho - \rho_0 - x}{\rho}\sin\theta_0$$

where $\pm$ is the sign of charge relative to the reference particle.

Then, the particle can reach the magnet exit $s_f = s + l_0$ iff $|f| < 1$ and the particle property (t_f, x_f, y_f, s_f) with $(E_f = E, p_{xf}, p_{yf}, p_{sf})$ in the beamline coordinate at the exit is exactly given by

$$\begin{aligned}
t_f &= t + \Delta t, \qquad \Delta t = \frac{1}{\omega_1}(\alpha + \theta_0 - \sin^{-1} f) \\
x_f &= x\cos\theta_0 - \rho_0(1 - \cos\theta_0) + \rho(\sqrt{1 - f^2} - \cos(\alpha + \theta_0)) \\
&= x\cos\theta_0 + 2(\rho - \rho_0)\sin^2\frac{\alpha + \theta_0}{2} + 2\rho_0 \sin\frac{\alpha}{2}\sin(\frac{\alpha}{2} + \theta_0) - \rho\frac{f^2}{1 + \sqrt{1 - f^2}} \\
y_f &= y + \frac{p_y}{E}\Delta t \\
s_f &= s + l \\
E_f &= E \\
p_{xf} &= f p_1 \\
p_{yf} &= p_y \\
p_{sf} &= \sqrt{1 - f^2}\, p_1 > 0
\end{aligned}$$

BMT equation

The spin equation of motion in an inertial frame is given by

$$\frac{d\boldsymbol{S}}{dt} = \left[(\gamma a + 1)\left(\frac{-e\boldsymbol{B}}{m\gamma}\right) - a(\gamma - 1)\left(\frac{-e\boldsymbol{B}}{m\gamma}\right)\cdot\boldsymbol{p}\frac{\boldsymbol{p}}{|\boldsymbol{p}|^2}\right]\times\boldsymbol{S}$$

to be compared with the equation for momentum

$$\frac{d\boldsymbol{p}}{dt} = \left(\frac{-e\boldsymbol{B}}{m\gamma}\right)\times\boldsymbol{p} = -\omega\boldsymbol{e}_y\times\boldsymbol{p}, \qquad \omega \equiv \frac{e m_0 \gamma_0}{e_0 m\gamma}\frac{v_0}{\rho_0}.$$

The BMT equation becomes simple if seen in a frame rotating with angular velocity ω around the y-axis: $\boldsymbol{e}_1 = \boldsymbol{e}_{10}\cos\omega t + \boldsymbol{e}_{30}\sin\omega t$, $\boldsymbol{e}_3 = -\boldsymbol{e}_{10}\sin\omega t + \boldsymbol{e}_{30}\cos\omega t$, $\boldsymbol{e}_2 = \boldsymbol{e}_y$.

$$\frac{d'\boldsymbol{S}}{dt} = \boldsymbol{\Omega}\times\boldsymbol{S}, \qquad \boldsymbol{\Omega} = \omega\left[-\gamma a\boldsymbol{e}_y + a(\gamma - 1)(\boldsymbol{e}_y\cdot\boldsymbol{p})\frac{\boldsymbol{p}}{|\boldsymbol{p}|^2}\right]$$

where the components of $\boldsymbol{\Omega}$ are constant in this frame. Thus, if the 3×3 rotation matrix corresponding to $\boldsymbol{\Omega}\Delta t$ (Δt already defined) is denoted by $M_{\boldsymbol{\Omega}\Delta t}$, then the final spin component in the beamline coordinate is

$$\begin{pmatrix} S_{xf} \\ S_{yf} \\ S_{sf} \end{pmatrix} = \begin{pmatrix} \cos\varphi_1 & 0 & \sin\varphi_1 \\ 0 & 1 & 0 \\ -\sin\varphi_1 & 0 & \cos\varphi_1 \end{pmatrix} M_{\boldsymbol{\Omega}\Delta t} \begin{pmatrix} S_x \\ S_y \\ S_s \end{pmatrix}, \qquad \varphi_1 = \omega\Delta t - \theta_0$$

5.5.4 Quadrupole Magnets

Since an exact solution is hard, the linear approximation employed in many accelerator codes is used. The edge effects are ignored. When a particle enters the magnet (characterized by the inverse focal length k_1 and the magnet length l) at (t, x, y, s) with (E, p_x, p_y, p_s) ($p_s > 0$), the values at the exit are computed by

$$\begin{aligned}
&x' = p_x/p_s, \qquad y' = p_y/p_s, \\
&x_f = \cos\theta\, x + l\frac{\sin\theta}{\theta}x' \qquad y_f = \cosh\theta\, x + l\frac{\sinh\theta}{\theta}y' \qquad \theta \equiv \sqrt{\pm\frac{p_0}{p}k_1 l} \\
&x'_f = -\frac{\theta}{l}\sin\theta\, x + \cos\theta\, x' \qquad y'_f = \frac{\theta}{l}\sinh\theta\, y + \cosh\theta\, y' \\
&p_{sf} = \sqrt{\frac{1 + x'^2 + y'^2}{1 + x_f'^2 + y_f'^2}}p_s, \qquad p_{xf} = x'_f p_{sf}, \qquad p_{yf} = y'_f p_{sf}
\end{aligned}$$

where $\pm$ is the sign of charge relative to the reference particle ($\pm k_1 > 0$ assumed. The opposite case is obvious.)

5.6 Luminosity

5.6.1 Luminosity Integration Algorithm

Let us denote the position-velocity distribution function of j-th beam (j=1,2) at time t by $\tilde{n}_j(\boldsymbol{r}, \boldsymbol{v}, t)$. It is normalized such that $\int \tilde{n}_j d\boldsymbol{r} d\boldsymbol{v}$ is the total number of particles in the j-th beam. The luminosity (per crossing) is in general given by

$$\mathcal{L} = \int \sqrt{(\boldsymbol{v}_1 - \boldsymbol{v}_2)^2 - (\boldsymbol{v}_1 \times \boldsymbol{v}_2)^2}\,\tilde{n}_1(\boldsymbol{r}, \boldsymbol{v}_1, t)\tilde{n}_2(\boldsymbol{r}, \boldsymbol{v}_2, t) d\boldsymbol{r} d\boldsymbol{v}_1 d\boldsymbol{v}_2 dt. \tag{5.48}$$

If all the particles in the j-th beam are ultrarelativistic and have almost the same velocity $\boldsymbol{v}_j$ ($|\boldsymbol{v}_j| \approx 1$), then the expression is simplified as

$$\mathcal{L} = (1 - \cos\phi) \int n_1(\boldsymbol{r}, t) n_2(\boldsymbol{r}, t) d\boldsymbol{r} dt \tag{5.49}$$

where ϕ is the polar angle between $\boldsymbol{v}_1$ and $\boldsymbol{v}_2$, and $n_j(\boldsymbol{r}, t) = \int \tilde{n}_j d\boldsymbol{v}$ is the number density of the j-th beam. **CAIN** uses this formula with $\phi = \pi$, ignoring the velocity distribution and the crossing angle.

The integration is done by introducing the time step size Δt, longitudinal slice width Δ_s, transverse mesh size Δ_x and Δ_y. Summing the number of particles in each bin, the luminosity is given by

$$\mathcal{L} = C \sum_{i_x, i_y, i_s, i_t} N^{(1)}_{i_x, i_y, i_s, i_t} N^{(2)}_{i_x, i_y, i_s, i_t} \Delta_x \Delta_y \Delta_s \Delta_t \tag{5.50}$$

where C is an appropriate normalization factor, and $N^{(j)}_{i_x, i_y, i_s, i_t}$ is the number of particles of the beam j in the bin (i_x, i_y, i_s, i_t). A problem is how to determine the transverse size of the bin (Δ_t and Δ_s is mainly determined by the dynamics — they are actually

Figure 5.2: Bin numbering for luminosity integration. Example with n=8 and the double-sized bin n=4.

42	43	46	47	58	59	62	63
40	41	44	45	56	57	60	61
34	35	38	39	50	51	54	55
32	33	36	37	48	49	52	53
10	11	14	15	26	27	30	31
8	9	12	13	24	25	28	29
2	3	6	7	18	19	22	23
0	1	4	5	16	17	20	21

10	11	14	15
8	9	12	13
2	3	6	7
0	1	4	5

specified by the user). If the bin is too large, detail of the distribution is lost, whereas if too small, statistical error becomes large because each bin will contain only a small number of macro-particles. **CAIN** adopts the following way.

At first, determine the size of the whole transverse region (w_x, w_y) such that most particles are contained there. Then, divide this region into as many bins $n \times n$ as allowed by the storage requirement (n must be a power of 2. **CAIN** uses $n = 128$.), and count the number of particles in each bin for both beams $N_k^{(j)}$ $(k = 0, 1, 2 \ldots, n^2 - 1)$.

If the number of macro-particles in any of the neighbouring 4 bins are less than some number N_{min} (**CAIN** adopts 5) for both beams, then sum these numbers and put the sum into a larger bin $(2\Delta_x, 2\Delta_y)$. (For the example in Fig.5.2, the sum of the bins 12, 13, 14, and 15 in the figure on the left corresponds to the bin 3 on the right.) Otherwise, add $N_k^{(1)} N_k^{(2)}$ into the luminosity sum. This doubling of the bin size is repeated so long as $N^{(j)} < N_{min}$. In order to make this algorithm efficient, the bin numbering system is a little complicated. Instead of using two indices (i_x, i_y), the bins are numbered as in Fig.5.2. With this numbering, the sum of neighbouring bins can be simply written as

$$N_{4k}^n + N_{4k+1}^n + N_{4k+2}^n + N_{4k+3}^n = N_k^{n/2} \tag{5.51}$$

where N_k^n is the number of particles in the k-th bin $(k = 0, 1, 2, \ldots, n^2 - 1)$ in $n \times n$ bin system.

5.6.2 Polarization

In the present version of **CAIN**, the particles are either photon or electron or positron. They all have two polarization eigenstates and can be specified by three real numbers, the Stokes parameter (ξ_1, ξ_2, ξ_3) for photons or the polarization vector $(\zeta_x, \zeta_y, \zeta_s)$ for electrons/positrons. Let us denote the three numbers in general by $\boldsymbol{s} \equiv (s_1, s_2, s_3)$.

Then, the crossection of a particular interaction integrated over a given final state (energy-momentum and polarization) is in general written in the form

$$\sigma = \sum_{i,j=0}^{3} \sigma_{i,j} s_i^{(R)} s_j^{(L)}, \tag{5.52}$$

where (R) and (L) represent the right and left-going particles and $s_0 = 1$ for notational convenience.

The number of events N during a beam collision is obtained by integrating eq.(5.52) with an appropriate factor over the momentum, the interaction volume and time:

$$N = \int d\boldsymbol{r}\, dt\, d\boldsymbol{p}^{(\boldsymbol{R})}\, d\boldsymbol{p}^{(\boldsymbol{L})} \sigma f \tag{5.53}$$

where f is the particle density functions with kinematic factors and is found in eq.(5.48). Since $\boldsymbol{s}^{(R)}$ and $\boldsymbol{s}^{(L)}$ depend on the particles in general, we get different coefficients from term to term of eq.(5.52). Thus, the number of events is written in the form

$$N = \sum_{i,j=0}^{3} \sigma_{i,j} \mathcal{L}_{i,j}, \qquad \mathcal{L}_{i,j} = \int d\boldsymbol{r}\, dt\, d\boldsymbol{p}^{(\boldsymbol{R})}\, d\boldsymbol{p}^{(\boldsymbol{L})} s_i^{(R)} s_j^{(L)} f \tag{5.54}$$

When the right and left-going beams are primary beams, the polarization is often uniform (*i.e.*, independent of energy-momentum and space-time) to a good approximation. In such a case $\mathcal{L}_{i,j}$ is simply given by $\mathcal{L} s_i^{(R)} s_j^{(L)}$, where $\mathcal{L} = \mathcal{L}_{00}$ is the total luminosity and $s_i^{(R,L)}$ is the polarization vectors of the beams. Then, the number of events is $\mathcal{L}\sigma$ where σ is given by eq.(5.54) with beam polarization value $s_i^{(R,L)}$ plugged-in.

Let us consider the helicity component in electron-electron collision. The helicity is approximately ζ_s and $-\zeta_s$ for right and left-going particles, respectively. The crosssections for four possible helicity combinations are

$$\begin{aligned} \sigma_{++} &= \sigma_{00} + \sigma_{30} - \sigma_{03} - \sigma_{33} \\ \sigma_{-+} &= \sigma_{00} - \sigma_{30} - \sigma_{03} + \sigma_{33} \\ \sigma_{+-} &= \sigma_{00} + \sigma_{30} + \sigma_{03} + \sigma_{33} \\ \sigma_{--} &= \sigma_{00} - \sigma_{30} + \sigma_{03} - \sigma_{33} \end{aligned} \tag{5.55}$$

The number of events is written as

$$N = \sigma_{++}\mathcal{L}_{++} + \sigma_{-+}\mathcal{L}_{-+} + \sigma_{+-}\mathcal{L}_{+-} + \sigma_{--}\mathcal{L}_{--} \tag{5.56}$$

with

$$\begin{aligned} \mathcal{L}_{++} &= \tfrac{1}{4}\left(\mathcal{L}_{00} + \mathcal{L}_{30} - \mathcal{L}_{03} - \mathcal{L}_{33}\right) \\ \mathcal{L}_{-+} &= \tfrac{1}{4}\left(\mathcal{L}_{00} - \mathcal{L}_{30} - \mathcal{L}_{03} + \mathcal{L}_{33}\right) \\ \mathcal{L}_{+-} &= \tfrac{1}{4}\left(\mathcal{L}_{00} + \mathcal{L}_{30} + \mathcal{L}_{03} + \mathcal{L}_{33}\right) \\ \mathcal{L}_{--} &= \tfrac{1}{4}\left(\mathcal{L}_{00} - \mathcal{L}_{30} + \mathcal{L}_{03} - \mathcal{L}_{33}\right) \end{aligned} \tag{5.57}$$

The total luminosity is $\mathcal{L}_{00}$. Note that the helicity is ξ_2 for photons. Thus, if both beams are photons, the above expression becomes

$$\begin{aligned} \mathcal{L}_{++} &= \tfrac{1}{4}\left(\mathcal{L}_{00} + \mathcal{L}_{20} + \mathcal{L}_{02} + \mathcal{L}_{22}\right) \\ \mathcal{L}_{-+} &= \tfrac{1}{4}\left(\mathcal{L}_{00} - \mathcal{L}_{20} + \mathcal{L}_{02} - \mathcal{L}_{22}\right) \\ \mathcal{L}_{+-} &= \tfrac{1}{4}\left(\mathcal{L}_{00} + \mathcal{L}_{20} - \mathcal{L}_{02} - \mathcal{L}_{22}\right) \\ \mathcal{L}_{--} &= \tfrac{1}{4}\left(\mathcal{L}_{00} + \mathcal{L}_{20} + \mathcal{L}_{02} + \mathcal{L}_{22}\right) \end{aligned} \tag{5.58}$$

The helicity luminosity is calculated by `LUMINOSITY` command by specifying the operand `HELICITY`. If you want all 16 combinations or the linear polarization effects, you need to specify `ALLPOL` operand. For electrons, the expression (5.57) is not exact because the helicity is defined as $\boldsymbol{\zeta}\cdot\boldsymbol{p}/\left|\boldsymbol{p}\right|$ rather than $\pm\zeta_s$.

5.7 Beam Field

One of the basic assumptions of **CAIN** is that the most particles in the beams have high energy and are almost either right- or left-going. This assumption leads to the following facts:

- A field due to a particle is almost concentrated in a transverse plane with the same s-coordinate of the particle because of the Lorentz contraction.
- If the electric field is $\boldsymbol{E}$, the magnetic field is given by $\boldsymbol{B} = \pm c\boldsymbol{e}_s \times \boldsymbol{E}$, where $\boldsymbol{e}_s$ is the unit vector along the s-axis, c the velocity of light, and the upper (lower) sign is for the field created by the right(left)-goin beam.

In contrast to **ABEL**, **CAIN** does not assume that all the particles have the above property. Some particles may have low energies and large angles with respect to the s-axis. **CAIN** will work, unless the sum of their weight becomes a significant fraction of the beam. The equation of motion under the Lorentz force is integrated with possible low energies and large angles taken into account.

The calculation of the beam electric field is done in the following way. First, cut the right(left)-going particles into longitudinal slices (the width Δs is defined by the parameter `Smesh`). Within each slice the following Poisson equation is solved.

$$\Delta\Phi(x,y) = 2\pi\rho_Q(x,y), \qquad \boldsymbol{E} = \frac{2mr_e}{\Delta s}\nabla\Phi, \tag{5.59}$$

where m is the electron mass in units of eV/c^2, r_e the classical electron radius in meters, $\rho_Q(x,y)$ is the charge (divided by the elementary charge) per unit transverse area, then $\boldsymbol{E}$ is given in units of V/m.

For each slice and for each of right- and left going beams, a region $(x_c \pm w_x/2, y_c \pm w_y/2)$ is selected, where (x_c, y_c) is the center-of-mass and (w_x, w_y) is the width determined by the input parameters. The field created by the particles outside this region is ignored. Let us name this region [O].

[O] Fast Fourier Transformation

In the region [O], the Poisson equation is solved using the FFT. Eq.(5.59) can formally be solved as

$$\Phi(\boldsymbol{r}) = \int_{-\infty}^{\infty} G(\boldsymbol{r}-\boldsymbol{r}')\rho_Q(\boldsymbol{r}')d\boldsymbol{r}', \qquad G(\boldsymbol{r}) = \log|\boldsymbol{r}|. \tag{5.60}$$

Divide this region by $n_x \times n_y$ grid. Within each cell (i,j) ($i = 1, \ldots n_x$, $j = 1, \ldots n_y$), the the density $\rho_Q(x,y)$ is approximated by $Q_{ij}/\Delta_x\Delta_y$, where $\Delta_x = w_x/n_x$, $\Delta_y = w_y/n_y$, and Q_{ij} is the total charge in the cell:

$$Q_{ij} = \int_{(x,y)\text{ in mesh }(ij)} \rho_Q(x,y)dxdy. \tag{5.61}$$

where (x_i, y_j) is the cell center coordinate. Then, eq.(5.60) becomes a sum over the cells.

$$\Phi(x_i, y_j) = \sum_{i',j'} G_{i-i',j-j'}Q_{i',j'}. \tag{5.62}$$

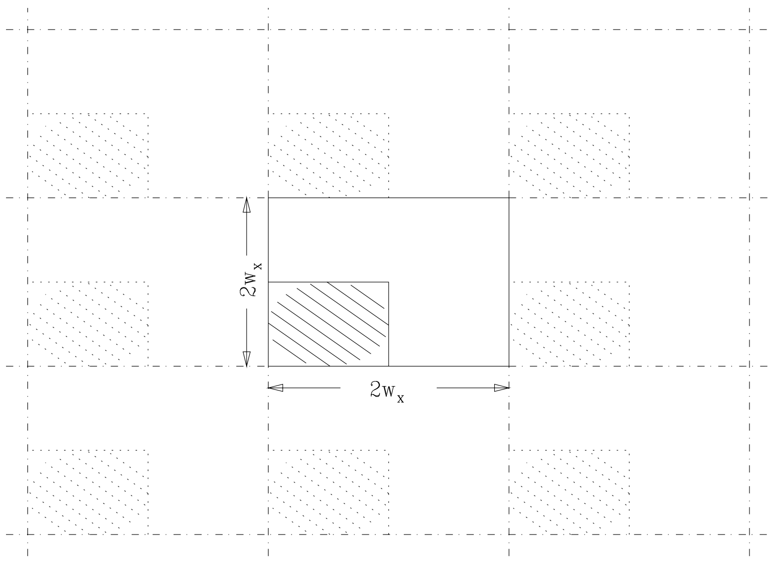


Figure 5.3: Doubled region for FFT. The solid frame indicates the doubled region for FFT and its left-bottom quadrant is the charge region $w_x \times w_y$. The region hatched by solid lines is the real charge region and that by dotted lines the ghost charge due to the periodicity of Fourier transformation.

The kernel matrix $G_{i,j}$ has to be calculated by taking average over the source cell:

$$G_{i,j} = \frac{1}{\Delta_x \Delta_y} \int_{(x,y) \text{ in cell } (i,j)} \frac{1}{2} \log[(x_i - x)^2 + (y_j - y)^2] dxdy \tag{5.63}$$

This averaging is important when Δ_x/Δ_y is far from unity. The convolution in eq.(5.62) can be done efficiently by using FFT.

However, if we apply FFT for the finite region (w_x, w_y) instead of the infinite region in eq.(5.60), we would be assuming a periodic charge distribution, i.e., the charge distribution in (w_x, w_y) is infinitely repeated. To avoid this problem, we use the following trick. First double the region to $(2w_x, 2w_y)$ by padding zero in the extended region and carry out FFT. This still means a periodic charge distribution as depicted in Fig.5.3. However, if we use the kernel matrix with zero padded in the extended region ($G_{i,j} = 0$ if $n_x < i \leq 2n_x$ or $n_y < i \leq 2n_y$), the field due to the ghost charges will never reach the real charge region because their horizontal(vertical) distance is larger than w_x (w_y). Thus, the potential Φ in the region (w_x, w_y) is calculated correctly although incorrect in the extended region.

The obtained values of the potential are those at cell centers. They are interpolated by 2-dimensional cubic spline and differentiated to get $\partial\Phi/\partial x$ and $\partial\Phi/\partial y$.

Outside the mesh region

When a charged particle gets out of the mesh region, the field created by it is ignored in **CAIN**2.35. However, the force by the other beam is taken into account even if the particle is outside the mesh region of the other beam. To this end, **CAIN**2.35 adopts three methods, namely, [A] direct Coulomb force by the charge distribution in the mesh, [B] harmonic expansion in polar coordinate, and [C] harmonic expansion in elliptic coordinate.

Let (w_x, w_y) be the total width of the mesh region. If it is close to a square, or more precisely, if $0.8 < w_x/w_y < 1.25$, the whole region is divided into three regions [O],[A],[B], as depicted in Fig.5.4a. If the mesh region is far from square, the whole region is divided

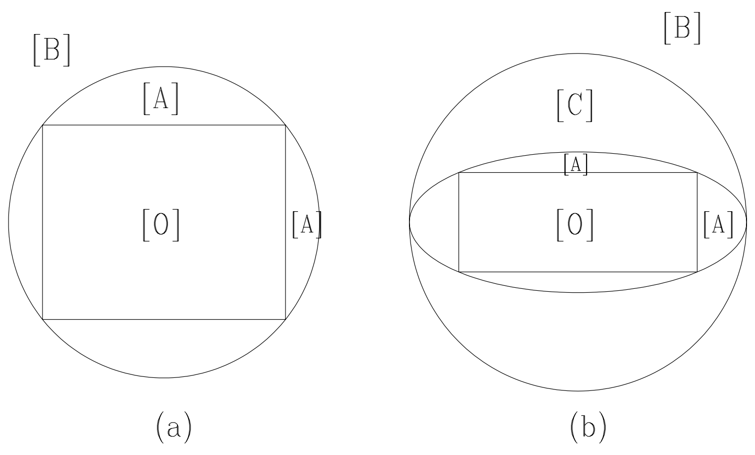


Figure 5.4: Regions for calculating the beam field

into four, [O],[A],[B],[C], as in Fig.5.4b. In the region [O] the mesh is used for calculating the field. In other regions, the methods mentioned above are used.

[A] Direct sum of Coulomb force

Since the sum is time consuming, this is used only in region [A], where two other methods fail to converge. The method is trivial and given by

$$\frac{\partial\Phi}{\partial x}+i\frac{\partial\Phi}{\partial y}=-\sum_{i,j}\frac{(x-x_i)+i(y-y_j)}{(x-x_i)^2+(y-y_j)^2}Q_{ij}, \tag{5.64}$$

However, this formula is not accurate when the bin size ratio Δ_x/Δ_y is far from unity. It is needed to take average over a bin when the bin is close to the field point (x,y). **CAIN** makes a table for the Coulomb force by a bin (Δ_x,Δ_y) for faster computation.

[B] Harmonic expansion in polar coordinate

In the region [B] the following formula is used.

$$\frac{\partial\Phi}{\partial x}-i\frac{\partial\Phi}{\partial y}=-\sum_{m=0}^{\infty}\left[\frac{r_0}{x+iy}\right]^{m+1}Q_m^B, \tag{5.65}$$

$$Q_m^B=\frac{1}{r_0}\int\rho_Q(x,y)\left[\frac{x+iy}{r_0}\right]^m dxdy. \tag{5.66}$$

Here, r_0 is arbitrary (introduced for avoiding overflow/underflow). The formula is valid for $x^2+y^2>r_{max}^2$, where $r_{max}=\sqrt{w_x^2+w_y^2}/2$ is the maximum radius of the mesh region.

[C] Harmonic expansion in elliptic coordinate

When $w_x > w_y$ (otherwise, exchange x and y), the elliptic coordinate (u, v) defined by

$$\begin{array}{ll} x = f\cosh u \cos v & \cosh(u+iv) = (x+iy)/f \\ y = f \sinh u \sin v & (u \geq 0, \quad 0 \leq v < 2\pi) \end{array} \tag{5.67}$$

is used. Here, f is chosen as

$$f = \sqrt{(w_x^2 - w_y^2)/2}. \tag{5.68}$$

The maximum of the radial-like coordinate u in the mesh region is

$$u_0 = \frac{1}{2}\log\frac{w_x + w_y}{w_x - w_y}, \tag{5.69}$$

which is taken at the four corners.

Then, the expansion of Φ is

$$\frac{\partial\Phi}{\partial x} - i\frac{\partial\Phi}{\partial y} = -\sum_{m=0}^{\infty} e^{-(m+1)(u-u_0+iv)} Q_m^C, \tag{5.70}$$

$$Q_m^C = \frac{2}{f}e^{-(m+1)u_0}\int \rho_Q(x,y)\frac{\sinh[(m+1)(u+iv)]}{\sinh(u+iv)}dxdy. \tag{5.71}$$

Actually, there is a finite relation between Q_m^B and Q_m^C:

$$Q_m^C = e^{-(m+1)u_0}\sum_{r=0}^{[m/2]}(-1)^r \begin{pmatrix} m-r \\ r \end{pmatrix}\left(\frac{2r_0}{f}\right)^{m-2r+1} Q_{m-2r}^B. \tag{5.72}$$

The formula converges if $u > u_0$, which corresponds to the region [C] (and [B]) in Fig.5.4b. The truncation of the series is defined by the operand `NMOM` of the command `BBFIELD` (common to the two types of expansions for simplicity).

5.8 Laser

5.8.1 Laser Geometry

Define a coordinate system attached to a laser. Let $\boldsymbol{e}^{(3)}$ be the unit vector along the direction of propagation, and introduce a unit vector $\boldsymbol{e}^{(1)}$ perpendicular to $\boldsymbol{e}^{(3)}$ and another unit vector $\boldsymbol{e}^{(2)} = \boldsymbol{e}^{(3)} \times \boldsymbol{e}^{(1)}$. The three vectors $(\boldsymbol{e}^{(1)}, \boldsymbol{e}^{(2)}, \boldsymbol{e}^{(3)})$ form an orthonormal frame. Define the components of these vectors in the original frame $(\boldsymbol{e}_x, \boldsymbol{e}_y, \boldsymbol{e}_s)$ as

$$\boldsymbol{e}^{(1)} = \begin{pmatrix} V_{11} \\ V_{21} \\ V_{31} \end{pmatrix}, \qquad \boldsymbol{e}^{(2)} = \begin{pmatrix} V_{12} \\ V_{22} \\ V_{32} \end{pmatrix}, \qquad \boldsymbol{e}^{(3)} = \begin{pmatrix} V_{13} \\ V_{23} \\ V_{33} \end{pmatrix}. \tag{5.73}$$

Then, $V = \{V_{ij}\}$ is a 3×3 orthogonal matrix. Let (ξ, η, ζ) be the spatial coordinate in this frame. Define the origin of (u_1, u_2, u_3) as the laser focus and let (x_0, y_0, s_0) be its coordinate in the original frame and t_0 the time when the laser pulse center passes the origin. Introduce a time coordinate τ whose origin is t_0. Now, the relation between (t, x, y, s) and (τ, ξ, η, ζ) is

$$\tau = t - t_0 \tag{5.74}$$

$$\begin{pmatrix} \xi \\ \eta \\ \zeta \end{pmatrix} = \begin{pmatrix} V_{11} & V_{21} & V_{31} \\ V_{12} & V_{22} & V_{32} \\ V_{13} & V_{23} & V_{33} \end{pmatrix} \begin{pmatrix} x - x_0 \\ y - y_0 \\ s - s_0 \end{pmatrix}. \tag{5.75}$$

A plane wave is written in the form, in the (τ, ξ, η, ζ) coordinate,

$$\boldsymbol{E} = \Re \boldsymbol{E}_0 e^{ik(\boldsymbol{n}\cdot\boldsymbol{r} - \tau)} \tag{5.76}$$

where $k = 1/\lambda\!\!\!^{-}_L = 2\pi/\lambda_L$ is the wave number, $\boldsymbol{n}$ is the unit vector along the propagation direction of the wave component, and $\boldsymbol{E}_0$ is a complex vector perpendicular to $\boldsymbol{n}$. A laser beam is considered to be a superposition of plane waves with slightly different $\boldsymbol{n}$ and k. If the distribution of $\boldsymbol{n}$ around $\boldsymbol{e}^{(3)}$ and that of k are Gaussian and if one ignores the $\boldsymbol{n}$ dependence of $\boldsymbol{E}_0$, the laser field can be approximated by

$$\boldsymbol{E} = \Re \boldsymbol{E}_0 e^{ik(\zeta - \tau)} \sqrt{A} e^{i\Phi}, \tag{5.77}$$

where

$$A(\tau, \xi, \eta, \zeta) = A_t(\tau - \zeta) A_s(\xi, \eta, \zeta) \tag{5.78}$$

$$A_t = \exp\left[-\frac{(\zeta - \tau)^2}{2\sigma_\tau^2}\right] \tag{5.79}$$

$$A_s = \frac{1}{\sqrt{d_1 d_2 [1 + (\zeta/\beta_1)^2][1 + (\zeta/\beta_2)^2]}} \exp\left[-\frac{\xi^2}{\lambda\!\!\!^{-}_L d_1(\beta_1 + \zeta^2/\beta_1)} - \frac{\eta^2}{\lambda\!\!\!^{-}_L d_2(\beta_2 + \zeta^2/\beta_2)}\right] \tag{5.80}$$

Here β_i $(i = 1, 2)$ is the Rayleigh length and σ_τ is the r.m.s. pulse length. d_i, which is unity to satisfy the Maxwell equation, is introduced for later use.

The wave front is given by the contour of $k\zeta + \Phi$. If one defines $\boldsymbol{n}'$ by $k\boldsymbol{n}' = \nabla(k\zeta + \Phi)$, $\boldsymbol{n}'$ is nearly a unit vector and approximated by

$$\boldsymbol{n}' = \frac{\boldsymbol{e}^{(3)} + c_1 \boldsymbol{e}^{(1)} + c_2 \boldsymbol{e}^{(2)}}{\sqrt{1 + c_1^2 + c_2^2}}, \tag{5.81}$$

$$c_1 = \frac{\xi\zeta}{d_1(\beta_1^2 + \zeta^2)}, \qquad c_2 = \frac{\eta\zeta}{d_2(\beta_2^2 + \zeta^2)}. \tag{5.82}$$

In **CAIN**, when the relevant particle is at (t, x, y, s), or at (τ, ξ, η, ζ) in laser coordinate, the laser field is considered to be locally a plane wave with the power density $A(\tau, \xi, \eta, \zeta) P_{peak}$, wave number k, and the propagation direction $\boldsymbol{n}'(\xi, \eta, \zeta)$.

There is some problem on the polarization because eq.(5.77) does not exactly satisfy the Maxwell equation. For simplicity, the basis $(\boldsymbol{e}'^{(1)}, \boldsymbol{e}'^{(2)}, \boldsymbol{e}'^{(3)})$ $(\boldsymbol{e}'^{(3)} = \boldsymbol{n}')$ for polarization is defined in the following manner: $\boldsymbol{e}'^{(1)}$ is the unit vector along $\boldsymbol{e}^{(1)} - (\boldsymbol{e}^{(1)} \cdot \boldsymbol{n}')\boldsymbol{n}'$ and $\boldsymbol{e}'^{(2)} = \boldsymbol{n}' \times \boldsymbol{e}'^{(1)}$. (This is irrelevant if only the longitudinal polarization is needed.)

The r.m.s. beam size at the position ζ is given. according to eq.(5.77), by

$$\sigma_i(\zeta) = \sqrt{d_i \frac{\lambda_L}{4\pi}\left(\beta_i + \frac{\zeta^2}{\beta_i}\right)}, \qquad (i = 1, 2). \tag{5.83}$$

To satisfy the Maxwell equation, d_i must be unity, but actual high-power lasers may not satisfy the above formula with $d_i = 1$. As a remedy to this problem, CAIN allows $d_i \neq 1$. The value of d_i is specified by the keyword `TDL` (times diffraction limit) in `LASER` command. However, this remedy is only a stopgap. It is presumably sufficient for linear Compton scattering but does not give a unique formula for nonlinear scattering. So, `TDL` must not be used with `LASERQED NPH`$\neq$0.

The formulas (5.77) to (5.83) apply to Gaussian beam. When the time (space) structure of the pulse is given by a file, A_t (A_s) is replaced by the values read from the file.

The Lorentz transformation is a little complicated because eq.(5.77) is far from a covariant form. The particle coordinates and the external fields are transformed immediately when `LORENTZ` command is invoked and the transformation parameters are forgotten. In the case of lasers, the transformation is not done immediately but instead the transformation parameters are stored. When the laser is called at every time step for each particle, the particle coordinates are Lorentz transformed back to the frame where the laser was defined, and the calculated parameters (A, ω', $\boldsymbol{e}'^{(1)}, \boldsymbol{e}'^{(2)}, \boldsymbol{e}'^{(3)}$) are transformed to the current Lorentz frame. Therefore, the Lorentz transformation is a little time-consuming.

Limitations

- When there are more than one lasers, there can be interference effects but these effects are not included in **CAIN**.
- When the pulse length is not very long compared with the wavelength, there is a spread in the laser frequency. In other words, the number of oscillations of the laser field felt by an electron is finite. This effect is not included.
- When the Rayleigh length is not very long compared with the wavelength, the laser wave front has a curvature. This is included only approximately by $\boldsymbol{n}'$ as in eq.(5.82). Laser field is treated locally as a plane wave.

Donuts-shaped Laser Beam

One can create a donuts-shaped beam, i.e., low intensity near the axis, by using the so-called axicon mirror/lens depicted in Fig.5.5

If the field at the entrance of the axicon is given by $\Phi_0(\tau-\zeta)e^{ik(\zeta-\tau)-r^2/4\sigma_0^2}$, the field after axicon is given by[7]

$$\Phi(r,\zeta,\tau-\zeta) = \Phi_0(\tau-\zeta)e^{ik(\zeta-\tau)}F(r,\zeta)$$

where

$$F(r,\zeta) = \frac{k}{\zeta+f}\exp\left\{i\frac{kr^2}{2(\zeta+f)}\right\}\int_b^a d\rho\sqrt{\rho(\rho-b)}\exp\left\{-\frac{(\rho-b)^2}{4\sigma_0^2} - i\frac{k\rho^2\zeta}{2(\zeta+f)f}\right\}J_0\left(\frac{kr\rho}{\zeta+f}\right)$$

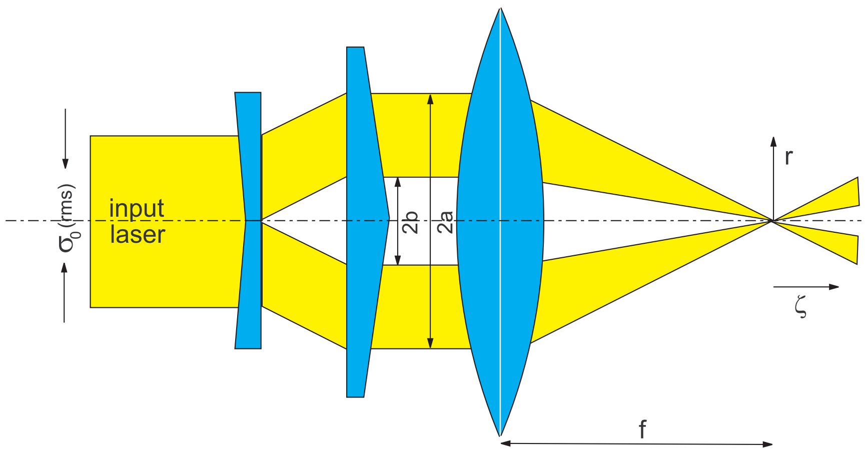


Figure 5.5: Geometry of axicon

Integration over the transverse plane ζ=const gives

$$\int |F(r,\zeta)|^2 \, 2\pi r dr = 2\pi\sigma_0^2 \left(1 - e^{-(a^2-b^2)/2\sigma_0^2}\right)$$

Note that in the absense of the axicon ($b=0$, $a=\infty$),

$$|F(r,\zeta)|^2 = \frac{\sigma_0^2}{\sigma(r,\zeta)^2} e^{-r^2/2\sigma(r,\zeta)^2}, \qquad \sigma(r,\zeta) = \frac{\zeta+f}{2k\sigma_0}\sqrt{1+\left(\frac{2k\sigma_0^2\zeta}{(\zeta+f)f}\right)^2}$$

$$\text{Rayleigh length} = \frac{f^2}{2k\sigma_0^2} = \frac{\lambda}{4\pi}\left(\frac{f}{\sigma_0}\right)^2$$

5.8.2 Linear Compton Scattering

When the parameter `NPH=0` is specified in `LASERQED` command, the formulas of linear Compton scattering are used.

Let us define the following variables in the rest frame of the initial electron:

ω,ω'	Initial (laser) and final energies of the photon.
$\boldsymbol{k}$,$\boldsymbol{k}'$	Initial (laser) and final momenta of the photon.
θ, ϕ	Polar and azimuthal scattering angle of the photon.
$d\Omega$	Solid angle $= \sin\theta d\theta d\phi = (m/\omega'^2)d\omega' d\phi$.
$\boldsymbol{\xi}^{(L)}$,$\boldsymbol{\xi}'^{(L)}$	Photon Stokes parameters before and after collision as defined in[3](page 361).[1] The definition of the axis is such that the first axis is parallel to $\boldsymbol{k}\times\boldsymbol{k}'$ and the third axis is $\boldsymbol{k}$ ($\boldsymbol{k}'$) for $\boldsymbol{\xi}^{(L)}$ ($\boldsymbol{\xi}'^{(L)}$).

[1] Actually, [3] adopts left-handed basis after collision so that the first and the second components of ξ' are $-\xi'_1$ and $-\xi'_2$ in [3]. Our $\xi'^{(L)}$ is not $(-\xi'_1, -\xi'_2, \xi'_3)$ but (ξ'_1, ξ'_2, ξ'_3).

The range of ω' is given by

$$\frac{\omega}{1+\lambda} \le \omega' \le \omega, \qquad \lambda = \frac{2\omega}{m} \tag{5.84}$$

The Compton relation is

$$\frac{1}{\omega'} - \frac{1}{\omega} = \frac{1-\cos\theta}{m}. \tag{5.85}$$

The crosssection is given by eq(87.22) in [3]. [2]:

$$\frac{d\sigma}{d\Omega} = \frac{1}{8}r_e^2\left(\frac{\omega'}{\omega}\right)^2 \times \Big[F_0 + F_3(\xi_3^{(L)} + \overline{\xi}_3^{'(L)}) + F_{11}\xi_1^{(L)}\overline{\xi}_1^{'(L)} + F_{22}\xi_2^{(L)}\overline{\xi}_2^{'(L)} + F_{33}\xi_3^{(L)}\overline{\xi}_3^{'(L)}$$
$$+(\boldsymbol{f}\cdot\boldsymbol{\zeta} + \boldsymbol{g}\cdot\overline{\boldsymbol{\zeta}}')\xi_2^{(L)} + (\boldsymbol{f}'\cdot\boldsymbol{\zeta} + \boldsymbol{g}'\cdot\overline{\boldsymbol{\zeta}}')\overline{\xi}_2^{'(L)} + \overline{\boldsymbol{\zeta}}'\cdot\mathcal{G}\cdot\boldsymbol{\zeta} + \dots\Big] \tag{5.86}$$

See Sec.5.2.2 for the meaning of the bars on $\boldsymbol{\zeta}$ and $\boldsymbol{\xi}^{'(L)}$. The omitted terms are products of three and four among $\boldsymbol{\zeta}$, $\overline{\boldsymbol{\zeta}}'$, $\boldsymbol{\xi}^{(L)}$, and $\overline{\boldsymbol{\xi}}^{'(L)}$. (Actually, we need the terms $\zeta\times\xi\times\overline{\zeta}'$ and $\zeta\times\xi\times\overline{\xi}'$ but they are not found in literature.) The functions introduced in the above expression are:

$$F_0 = \frac{\omega}{\omega'} + \frac{\omega'}{\omega} - \sin^2\theta, \qquad F_3 = \sin^2\theta, \tag{5.87}$$

$$F_{11} = 2\cos\theta, \qquad F_{22} = \left(\frac{\omega}{\omega'} + \frac{\omega'}{\omega}\right)\cos\theta, \qquad F_{33} = 1+\cos^2\theta, \tag{5.88}$$

$$\boldsymbol{f} = -\frac{1}{m}(1-\cos\theta)(\boldsymbol{k}\cos\theta + \boldsymbol{k}'), \qquad \boldsymbol{f}' = -\frac{1}{m}(1-\cos\theta)(\boldsymbol{k} + \boldsymbol{k}'\cos\theta), \tag{5.89}$$

$$\begin{bmatrix}\boldsymbol{g}\\ \boldsymbol{g}'\end{bmatrix} = \begin{bmatrix}\boldsymbol{f}\\ \boldsymbol{f}'\end{bmatrix} + \frac{\sin^2\theta}{m}\frac{\omega+\omega'}{\omega-\omega'+2m}(\boldsymbol{k}-\boldsymbol{k}'), \tag{5.90}$$

$$\begin{aligned}\mathcal{G} = \; & I(1+\cos^2\theta + \frac{\omega-\omega'}{2m}\sin^2\theta) + \frac{\omega+\omega'}{2m}(1+\cos\theta)(\boldsymbol{n}'\otimes\boldsymbol{n} - \boldsymbol{n}\otimes\boldsymbol{n}') \\ & -\frac{\omega-\omega'}{2m}[(\boldsymbol{n}+\boldsymbol{n}')\otimes(\boldsymbol{n}+\boldsymbol{n}') + (\boldsymbol{n}\times\boldsymbol{n}')\otimes(\boldsymbol{n}\times\boldsymbol{n}')] \\ & +\frac{\omega-\omega'}{m(\omega-\omega'+2m)}(1+\cos\theta)(\boldsymbol{k}-\boldsymbol{k}')\otimes(\boldsymbol{n}+\boldsymbol{n}') \\ & -\frac{\sin^2\theta + 2\cos\theta}{m(\omega-\omega'+2m)}(\boldsymbol{k}-\boldsymbol{k}')\otimes(\boldsymbol{k}-\boldsymbol{k}').\end{aligned} \tag{5.91}$$

These formulas are used in their exact forms in **CAIN**.

Summation over the final polarization and the azimuthal angle ϕ gives the differential crosssection with respect to the final photon energy ω'. Introducing the variables $\overline{z}$ inplace of ω' by

$$\overline{z} = z/L_\lambda \quad (0 \le \overline{z} \le 1), \qquad z = \log(\omega/\omega'), \qquad L_\lambda \equiv \log(1+\lambda), \tag{5.92}$$

$$\cos\theta = 1 - 2(e^z - 1)/\lambda, \tag{5.93}$$

[2]The term $\mathcal{G}$ is given in eq(4.6) in [4]

we write the differential crossection as

$$\frac{d\sigma}{d\overline{z}} = 4\pi r_e^2 \frac{L_\lambda}{\lambda} F(\overline{z}), \tag{5.94}$$

where

$$F(\overline{z}) = \frac{1}{2L_\lambda}\left[1 + e^{-2z} - e^{-z}\sin^2\theta - h\cos\theta(1-e^{-2z})\right], \tag{5.95}$$

$$h = \frac{\boldsymbol{k}}{\omega}\cdot\boldsymbol{\zeta}\,\xi_2^{(L)}. \tag{5.96}$$

Note that ξ_3 does not appear here because it is based on the scattering plane and, therefore, disappears after integration over the azimuthal angle. The function $F(\overline{z})$ satisfies $0 \le F \le 1$ for any $\overline{z}$ and λ and is $O(1)$ except when h is close to $+1$ and λ is extremely large. The Function $F(\overline{z})$ is plotted in Fig.5.6.

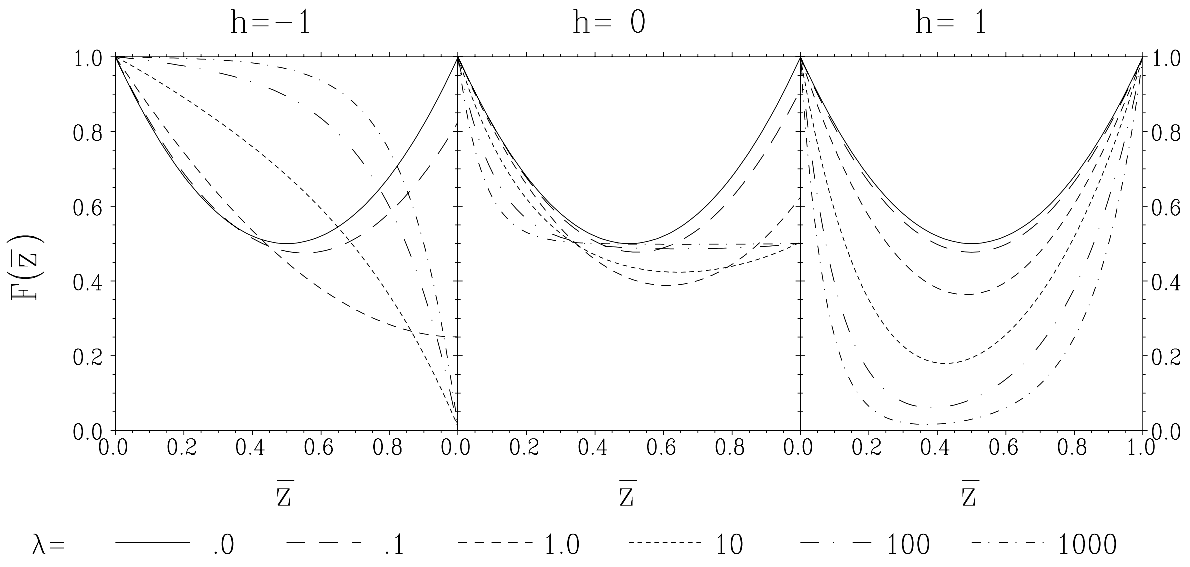


Figure 5.6: Function $F(\overline{z})$ for $h = 0, \pm 1$ for various values of λ

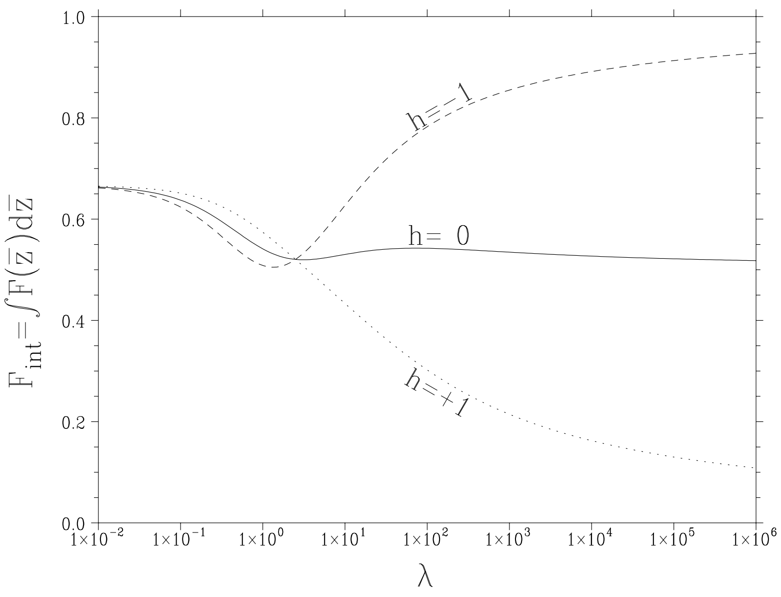
The total crosssection for given initial momenta and polarizations is given by

$$\sigma_{tot} = 4\pi r_e^2 \frac{L_\lambda}{\lambda} F_{int}(\lambda), \tag{5.97}$$

$$F_{int}(\lambda) = \frac{1}{2L_\lambda}\left\{\left(1-\frac{4}{\lambda}-\frac{8}{\lambda^2}\right)L_\lambda + \frac{1}{2}+\frac{8}{\lambda}-\frac{1}{2(1+\lambda)^2} + h\left[-\left(1+\frac{2}{\lambda}\right)L_\lambda + 2 + \frac{1}{2(1+\lambda)^2}\right]\right\} \tag{5.98}$$

Let us driefly describe the algorithm of event generation.

Figure 5.7: Function $F_{int}(\lambda)$ for $h = 0, \pm 1$ as a function of λ. F_{int} is less than unity and is $O(1)$ unless h is close to $+1$ and λ is extremely large.



1. Compute the total event rate P_0 in the given time interval using σ_{tot} without the factor F_{int}. Since $F_{int} \leq 1$, this is an over estimation of the rate. If P_0 is too large, divide the time interval by an integer N and repeat the following procedure N times.

2. Generate a random number r_1 uniform in (0,1). Reject if $r_1 \geq P_0$.

3. Compute F_{int} and multiply it to P_0. Reject if still $r_1 \geq P_0$. Otherwise accept. Note that the Lorentz transformation of $\boldsymbol{\xi}^{(L)}$ is not needed for the computation of h because ξ_2 is Lorentz invariant. Also note that input $\boldsymbol{\zeta}$ is defined already in the rest frame of electron. Only the Lorentz transformation of $\boldsymbol{k}$ is needed.

4. Generate two random numbers $\overline{z}$ and r_2 in (0,1). Repeat this step until $r_2 < F(\overline{z})$ is satisfied. Once or twice repetition is normally enough unless h is close to $+1$ and λ is very large.

5. Compute ω' from $\overline{z}$. Generate the azimuthal angle, compute the final polarization if needed, and go back to the laboratory frame. In this step many Lorentz transformations are needed.

5.8.3 Compton Process in a Strong Laser Field

When the laser field is strong, the simple formulas of Compton can nolonger be used. The laser field strength is characterized by the parameter

$$\xi = \frac{e\sqrt{-\langle A^\mu A_\mu \rangle}}{m} = \frac{\lambdabar_L}{m}\sqrt{\mu_0 c P} \tag{5.99}$$

Here, A^μ is the 4-vector potential of the laser field and $\langle\ \rangle$ indicates the average over the phase. $\lambdabar_L$ is the laser wavelength $/(2\pi)$ in meter, m the electron rest mass in eV/c^2, c the velocity of light in m/s, $\mu_0 = 4\pi \times 10^{-7}$, and P the power density in Watt/m^2.

When $\xi \ll 1$, the well-knwon formulas are enough but as ξ becomes large, the probability of absorbing more than one photon in the laser field cannot be ignored. When $\xi \gg 1$, the constant-field approximation[3] becomes good. If $\xi < O(1)$, the expansion in terms of the number of absorbed photons, n, shows good convergence. The expansion takes a relatively simple form when the laser is circularly polarized by 100%. Old versions treat only circular polarization. 100% linear polarization has also been accepted since **CAIN**2.35. **However, for linear laser polarization only the Compton scattering is ready, i.e., the Breit-Wheeler process is not ready. Moreover, the electron spin is ignored in the case of linear laser polarization.**

5.8.3.1 Kinematics of Nonlinear Compton Process

We work in the 'head-on frame' where the laser beam and the electron collide head-on. In the following, $\approx$ is an approximation that the electron is ultra-relativistic in this frame. We use the following notation.

p,k_L,p',k	4-momenta of initial electron, laser photon, final electron and emitted photon, respectively.
$\mathcal{E}$,ω_L,$\mathcal{E}'$,ω	Energies of initial electron, laser photon, final electron and emitted photon, respectively.
λ	Laser energy parameter: $\lambda = 2k_L{\cdot}p/m^2 \approx 4\omega_L\mathcal{E}/m^2$.
n	Number of absorbed laser photon.

When an electron with 4-momentum p^μ goes adiabatically into a strong laser field characterized by ξ, it behaves as if its momentum (called 'quasi-momentum) is

$$q = p + \frac{m^2\xi^2}{2p{\cdot}k_L}k_L, \qquad (\ q^2 = (1+\xi^2)m^2\ ). \tag{5.100}$$

The momentum conservation of the process of n photon absorption is therefore takes the form $q^\mu + nk_L^\mu = q'^\mu + k^\mu$ where q'^μ is defined like q^μ with p replaced by p'.

x	$x = (k{\cdot}k_l)/(p{\cdot}k_L) \approx \omega/\mathcal{E}$, $(0 < x < 1)$
v	$v = x/(1-x)$, $x = v/(1+v)$. $(0 < v < \infty)$. $dv/(1+v)^2 = dx$.
v_n	Maximum v for given n: $v_n = n\lambda/(1+\xi^2)$.
x_n	Maximum x for given n: $x_n = v_n/(1+v_n) = n\lambda/(1+\xi^2+n\lambda)$.
$\mathcal{E}_{eff}$	$= q^0 = \mathcal{E} + (\xi^2/\lambda)\omega_L$ is the effective energy of initial electron in the laser field.
Θ	Lorentz invariant variable defined by

$$\Theta \equiv \sqrt{n\lambda\frac{1-x}{x} - (1+\xi^2)} = \sqrt{(1+\xi^2)\left(\frac{v_n}{v} - 1\right)} \tag{5.101}$$

The photon scattering angle θ_γ ($\theta_\gamma = 0$ for backward) in the head-on frame can be written as $\theta_\gamma \approx m\Theta/\mathcal{E}$.

[3]This should be treated by the `CFQED` command. If users want, `EXTERNALFIELD` command will be rewritten so as to accept varying fields.

Once x and n are given, the final momentuma are given in the head-on frame by

$$k^\mu = xp^\mu + \left[n(1-x) - \frac{2+\xi^2}{\lambda} x \right] k_L^\mu + mx\Theta \left(e_1^\mu \cos\phi + e_2^\mu \sin\phi \right) \tag{5.102}$$

$$p'^\mu = p^\mu + \left(n - \frac{\xi^2}{\lambda} \frac{x}{1-x} \right) k_L^\mu - k^\mu . \tag{5.103}$$

Here, e_1^μ is a spatial vector perpendicular to $\boldsymbol{p}$ (and to $\boldsymbol{k}_L$) [4] and e_2^μ is defined by

$$e_2^\mu \equiv \frac{\epsilon_{\mu\nu\alpha\beta} e_1^\nu p^\alpha k_L^\beta}{p \cdot k_L} = \begin{bmatrix} 0 \\ \boldsymbol{e}_1 \times (\boldsymbol{p}\omega_L - \mathcal{E}\boldsymbol{k}_L)/p \cdot k_L \end{bmatrix} , \tag{5.104}$$

where $\epsilon^{\mu\nu\alpha\beta}$ is the completely anti-symmetric tensor ($\epsilon^{0123} = +1$). ϕ is the azimuthal scattering angle in the head-on frame (its distribution is uniform in $[0,2\pi]$ for circularly polarized laser).

5.8.3.2 Case of Circularly Polarized Lasers

Transition rate

The transition rate per unit time is given by

$$W = \frac{\alpha m^2 \xi^2}{4q_0} \sum_{n=1}^{\infty} \int_0^{x_n} dx \left[(1+h_e \overline{h}_{e'}) F_{1n} + h_L (h_e + \overline{h}_{e'}) F_{2n} + h_e \overline{h}_{e'} F_{5n} + \overline{h}_\gamma (h_L F_{3n} + h_e F_{4n}) \right] \tag{5.105}$$

h_L	Laser helicity (-1 or $+1$)
h_e,$h_{e'}$	Initial and final electron helicities ($-1 \leq h_e \leq 1$)
h_γ	Final photon helicity
$\overline{h}_{e'}$,$\overline{h}_\gamma$	'Detector helicity' of the final particles. See section 65 of [3].

The total transition rate is (sum over $\overline{h}_{e'}$ and $\overline{h}_\gamma$)

$$W = \frac{\alpha m^2 \xi^2}{4q_0} \sum_{n=1}^{\infty} \int_0^{x_n} dx \left[F_{1n} + h_L h_e F_{2n} \right] \tag{5.106}$$

The functions F_{kn} are defined by

$$F_{1n} = -\frac{1}{\xi^2} J_n^2 + \frac{1}{2} \left[1 + \frac{v^2}{2(1+v)} \right] (J_{n-1}^2 + J_{n+1}^2 - 2J_n^2) \tag{5.107}$$

$$F_{2n} = \left(\frac{1}{2} - \frac{v}{v_n} \right) \frac{v(1+v/2)}{1+v} (J_{n-1}^2 - J_{n+1}^2) \tag{5.108}$$

$$F_{3n} = \left(\frac{1}{2} - \frac{v}{v_n} \right) \left[1 + \frac{v^2}{2(1+v)} \right] (J_{n-1}^2 - J_{n+1}^2) \tag{5.109}$$

$$F_{4n} = -\frac{v}{1+v} \frac{1}{\xi^2} J_n^2 + \frac{1}{2} \frac{v(1+v/2)}{1+v} (J_{n-1}^2 + J_{n+1}^2 - 2J_n^2) \tag{5.110}$$

$$F_{5n} = -\frac{v^2}{1+v} \frac{1}{\xi^2} J_n^2 \tag{5.111}$$

[4]The direction of e_1^μ is arbitrary so long as perpendicular to $\boldsymbol{p}$. For convenience we choose the polarization plane of the laser for the case of linear polarization.

J_n's are Bessel functions with the argument z_n defined by

$$z_n = 2n\frac{\xi}{\sqrt{1+\xi^2}}\sqrt{\frac{v}{v_n}\left(1-\frac{v}{v_n}\right)} \tag{5.112}$$

F_{1n}, F_{2n}, F_{3n}, F_{4n} are identical to D_{1n}, D_{2n}, N_{1n}, N_{1n} divided by $8\xi^2$ in Tsai's paper[5], although the expressions in his paper look much more complicated.

Algorithm of Event Generation

When the command `LASERQED` is invoked, **CAIN** creates a table of the functions F_{1n} and F_{2n}. They are three-argument functions. In the program, ξ^2, λ, and $y = v/v_n$ $(0 \le y \le 1)$ are used as the independent variables instead of (ξ, λ, x). The functions F_{kn} (k=1,2) are stored in a 5-dimensional array `FF`(k,n,i,j,l) (n=1,`NPH`), (i=0,`NY`), (j=0,`NXI`), (l=0,`NLAMBDA`). (The numbers `NPH` etc. are specified by the `LASERQED` command.) The integral over y from 0 to y_i is stored in `FINT`(k,n,i,j,l). The integral over the full range $0 \le y \le 1$ is then `FINT`(k,n,`NY`,j,l) For integration, the trapezondal rule is used, which means the function F_{kn} is approximated by a piecewise linear function. The sum of `FINT`(k,n,`NY`,j,l) over n is stored in `FALL`(k,j,l).

For a given initial condition, calculate the parameters ξ^2 and λ and find `FALL`($*$) by 2-dimensional interpolation. (The asterisk $*$ indicates the appropriate sum over the initial polarization, *i,e.,* `FALL`($*$)=`FALL`$_{k=1}+h_L h_e$`FALL`$_{k=2}$). Then, calculate the total probability P (eq.(5.106) times the time interval `DT`):

$$P = C_0 \times \texttt{FALL}(*), \qquad C_0 = \frac{\alpha m^2\xi^2\Delta t}{\mathcal{E}_{eff}}. \tag{5.113}$$

Generate a uniform random number r_1 in the interval (0,1). If $r_1 < P$, decide to emit a photon and, otherwise reject.

If rejected, the helicity of the electron should be changed, according to eq.(5.12), to

$$h_{e,new} = \frac{h_e(1-P_1) - h_L P_2}{1-P}, \qquad P_k = C_0 \times \texttt{FALL}_k \qquad (k=1,2). \tag{5.114}$$

If accepted, decide how many laser photons to absorb. To do so, sum up `FINT`($*$,n,`NY`,j,l) from $n=1$ to $n=n_1$ until the sum becomes larger than r_1. Then, n_1 will be the number of photons.

Once n_1 is determined, the photon energy is determined by

$$\int_0^y dy\,\texttt{FF}(*) = r_2 \times \texttt{FINT}(*)$$

where r_2 is another uniform random number. The left hand side is known for the mesh point of y (*i.e.,* `FINT`(k,n,`NY`,j,l)). Since we approximate F_{kn} by a piecewise linear function of y, the left hand side is a quadratic function between successive y's. Thus, inverse interpolation with respect to i by solving a quadratic equation gives the photon energy to be emitted.

The helicities of the final photon and electron are calculated from

$$h_\gamma = \frac{h_L F_{3n} + h_e F_{4n}}{F_{1n} + h_L h_e F_{2n}} \tag{5.115}$$

$$h_{e'} = \frac{h_e(F_{1n}+F_{5n}) + h_L F_{2n}}{F_{1n} + h_L h_e F_{2n}} \tag{5.116}$$

for $n = n_1$. This is done by directly calling a Bessel function routine without using a table.

5.8.3.3 Case of Linearly Polarized Lasers

Transition Rate

When the laser is linearly polarized by 100%, the transition probability for unpolarized electron summed over final electron polarization is

$$W = \frac{\alpha m^2 \xi^2}{2E^2} \sum_{n=1}^{\infty} \int_0^{\omega_n} d\omega \int_0^{2\pi} \frac{d\phi}{2\pi} \left[f_{0n} + f_{1n}\overline{\xi}_1 + f_{3n}\overline{\xi}_3 \right] . \tag{5.117}$$

$\overline{\xi}_3$, $\overline{\xi}_1$	The Stokes parameter of the final photon (based on the polarization plane of the initial laser).
ϕ	The azimuthal scattering angle ($\phi = 0$ is the polarization plane of the initial laser).

The functions $f_{in}(x,\phi)$ $(i = 0, 1, 3)$ are given by

$$\begin{aligned}
f_{0n} &= -\frac{|A_n^{(0)}|^2}{\xi^2} + \left(1 - x + \frac{1}{1-x}\right)\left[|A_n^{(1)}|^2 - A_n^{(0)}A_n^{(2)}\right], & (5.118)\\
f_{1n} &= \frac{|A_n^{(0)}|^2}{\xi^2}\Theta^2 \sin 2\phi + 2\sqrt{2}\frac{\Theta}{\xi}\sin\phi A_n^{(0)}A_n^{(1)}, & (5.119)\\
f_{3n} &= -(1 + 2\Theta^2 \sin^2\phi)\frac{|A_n^{(0)}|^2}{\xi^2} + 2\left[|A_n^{(1)}|^2 - A_n^{(0)}A_n^{(2)}\right]. & (5.120)
\end{aligned}$$

where $A_n^{(s)}$ is defined by

$$A_n^{(s)} = \oint \frac{d\phi}{2\pi} \cos^s \phi e^{i[n\phi - \alpha_1 \sin\phi + (\alpha_2/2)\sin(2\phi)]}, \qquad (s = 0, 1, 2) \tag{5.121}$$

with arguments

$$\begin{aligned}
\alpha_1 &= e\left(\frac{a\cdot p}{k\cdot p} - \frac{a\cdot p'}{k\cdot p'}\right) = -2\sqrt{2}\,\frac{\xi}{\Lambda}\frac{x\Theta}{1-x}\cos\phi,\\
\alpha_2 &= \frac{e^2 a^2}{4}\left(\frac{1}{k\cdot p} - \frac{1}{k\cdot p'}\right) = \frac{\xi^2}{\Lambda}\frac{x}{1-x}.
\end{aligned}$$

(a^μ is the vector potential A^μ with the phase factor $e^{-ik_L x}$ omitted.)

The algorithm is more complicated than the circular polarization case because of one more variable ϕ but the method is similar.

5.8.4 Breit-Wheeler Process in a Strong Laser Field

5.8.4.1 Kinematics of Nonlinear Breit-Wheeler Process

k,k_L,p,p'	4-momenta of the initial photon, laser photon, final electron and positron.
ω,ω_L,ϵ,ϵ'	Energies of the initial photon, laser photon, final electron and positron.
ξ	Laser intensity parameter
η	Laser energy parameter: $\eta = k{\cdot}k_L/2m^2 \approx \omega_L\omega/m^2$.
n	Number of absorbed laser photons
x	$=p{\cdot}k_L/k{\cdot}k_L \approx \epsilon/\omega$. $(0 < x < 1)$
u	$u = 1/[4x(1-x)]$, $x = \frac{1}{2}[1 \pm \sqrt{1-1/u}]$, $(1 < u < \infty)$.
u_n	Maximum u for given n: $u_n = n\eta/(1+\xi^2)$.
x_n	$x_n = \frac{1}{2}[1 - \sqrt{1-1/u_n}]$
Θ	Lorentz invariant variable defined by

$$\Theta \equiv \sqrt{4\eta n x(1-x) - (1+\xi^2)} = \sqrt{(1+\xi^2)\left(\frac{u_n}{u} - 1\right)} \qquad (5.122)$$

The polar angle of final electron is $\theta_e \approx m\Theta/\epsilon$ in the head-on frame.

For given x and n, the final momenta are given by

$$p = xk + \left[n(1-x) - \frac{4\xi^2}{\eta x}\right] k_L + m\Theta(e_1 \cos\phi + e_2 \sin\phi), \qquad (5.123)$$

$$p' = (1-x)k + \left[nx - \frac{4\xi^2}{\eta(1-x)}\right] k_L - m\Theta(e_1 \cos\phi + e_2 \sin\phi). \qquad (5.124)$$

Here, ϕ is the azimuthal scattering angle in a head-on frame. e_1^μ is a spatial vector perpendicular to $\boldsymbol{k}_L$ (and k) and e_2^μ is defined by

$$e_2^\mu \equiv \frac{\epsilon_{\mu\nu\alpha\beta} e_2^\nu k^\alpha k_L^\beta}{k{\cdot}k_L} = \begin{bmatrix} 0 \\ \boldsymbol{e}_1 \times (\boldsymbol{k}\omega_L - \omega\boldsymbol{k}_L)/k{\cdot}k_L \end{bmatrix}. \qquad (5.125)$$

These vectors satisfy

$$k{\cdot}e_j = k_L{\cdot}e_j = 0, \quad e_j{\cdot}e_j = -1, \quad (j = 1,2), \qquad e_1{\cdot}e_2 = 0. \qquad (5.126)$$

5.8.4.2 Case of Circularly Polarized Lasers

Transition Rate

Total number of pair electrons per unit time summed over the positron polarization is

$$W = \frac{\alpha m^2 \xi^2}{2\omega} \sum_{n=1,\ n>(1+\xi^2)/\eta}^{\infty} \int_{x_n}^{1-x_n} dx [G_{1n} + h_L h_\gamma G_{3n} + \overline{h}_e (h_L G_{2n} + h_\gamma G_{4n})] \quad (5.127)$$

h_γ	Initial photon helicity

h_L	Laser helicity
h_e	Final electron helicity
$\overline{h}_e$	'Detector' helicity of the final electron.
z_n	The argument of the Bessel functions in the following expressions: $z_n = 2n\frac{\xi}{\sqrt{1+\xi^2}}\sqrt{\frac{u}{u_n}\left(1-\frac{u}{u_n}\right)}$

Sum over final electron polarization gives

$$W = \frac{\alpha m^2\xi^2}{\omega} \sum_{n=1,\ n>(1+\xi^2)/\eta}^{\infty} \int_{x_n}^{1-x_n} dx[G_{1n} + h_L h_\gamma G_{3n}] \tag{5.128}$$

The functions G_{kn}'s are defined by

$$G_{1n} = \frac{1}{\xi^2}J_n^2 + \frac{1}{2}(2u-1)(J_{n-1}^2 + J_{n+1}^2 - 2J_n^2) \tag{5.129}$$

$$G_{2n} = (1-2x)2u\left(\frac{1}{2} - \frac{u}{u_n}\right)(J_{n-1}^2 - J_{n+1}^2) \tag{5.130}$$

$$G_{3n} = -(2u-1)\left(\frac{1}{2} - \frac{u}{u_n}\right)(J_{n-1}^2 - J_{n+1}^2) \tag{5.131}$$

$$G_{4n} = \frac{1}{x\xi^2}J_n^2 - u(1-2x)(J_{n-1}^2 + J_{n+1}^2 - 2J_n^2) \tag{5.132}$$

These formulas can be obtained from those of F_{kn} for the Compton process by the replacement

$$\begin{array}{lll} \omega \rightarrow -\omega, & \quad & \mathcal{E} \rightarrow -\epsilon \\ h_\gamma \rightarrow -h_\gamma, & & h_e \rightarrow -h_e. \end{array} \tag{5.133}$$

This implies

$$\begin{array}{lll} v \rightarrow -1/(1-x), & \quad & v_n \rightarrow -4xu_n \\ v/v_n \rightarrow u/u_n, & & z_n \rightarrow z_n. \end{array} \tag{5.134}$$

For convenience, we have changed the sign as

$$F_{1n} \rightarrow -G_{1n}, \quad F_{2n} \rightarrow +G_{2n}, \quad F_{3n} \rightarrow +G_{3n}, \quad F_{4n} \rightarrow -G_{4n}. \tag{5.135}$$

Algorithm of Event Generation

Because of the threshold behavior of n photon process, the algorithm is slightly different from that for the Compton process. G_{kn} are 3-argument functions. In order to avoid discontinuities due to the n-photon threshold, following variables (in addition to ξ^2) are used in **CAIN** as the independent variables instead of (ξ, η, x):

q	Defined by $q = \eta/(1+\xi^2)$. The n-photon threshold is given by $q \geq 1/n$.

y Defined by $x = \frac{1}{2} - \frac{1}{2}\sqrt{1-1/u_n}y$, i.e., $y = (1-2x)/\sqrt{1-1/u_n}$. The range $0 \le y \le 1$ represents electrons with energy $\le \omega/2$ and $-1 \le y \le 0$ those $\ge \omega/2$. Since G_{1n} and G_{3n} are even functions of y, only the part $y \ge 0$ is tabulated.

The mesh for q cannot be equally spaced. Select q's such that all the threshold points $1/n$ ($n = n_{min}, n_{min}+1, \ldots$ `NPH`, n_{min} = integer part of 1/`ETAMAX`) are included, that q are equally spaced between successive thresholds, and that the spaces are not very diffrent. Thus, the total number of q's may not exactly equal to `NQ`.

The functions stored in the array `GG`(k,n,i,j,l) are G_{1n} (k=1) and G_{3n} (k=2). The integral `GINT`(k,n,i,j,l) is calculated as in the Compton case. The sum of `GINT`(k,n,`NY`,j,l) over n does not make sense because the result would be a quite discontinuos function of ξ and q, which makes the interpolation inaccurate. However, for given q, the sum from $n = n(q)$ to `NPH` is continuous where $n(q)$ is the minimum integer which does not exceed $1/q$. Thus, the sum over $n = n$ to `NPH` is stored in `GALL`(k,n,j,l).

For given initial condition, calculate ξ and q. Then, interpolate `GALL`(k,$n(q)$,j,l) for j and l and calculate the total probability P by summing for appropriate polarization:

$$P = C_0 \times \mathtt{GALL}(*), \qquad \mathtt{GALL}(*) = \mathtt{GALL}_{k=1} + \mathtt{GALL}_{k=2}, \qquad C_0 = \frac{\alpha m^2 \xi^2 \Delta t}{\omega}. \tag{5.136}$$

Generate a uniform random number r_1 and reject an event if $r_1 \ge P$. In this case the helicity of the photon should change to

$$h_{\gamma,new} = \frac{h_\gamma(1-P_1) - h_L P_2}{1-P}, \qquad P_k = C_0 \times \mathtt{GALL}_k, \quad (k = 1,2). \tag{5.137}$$

If $r_1 < P$, decide to create a pair. The number of laser photons to absorb is determined from `GINT`($*$,n,`NY`,j,l) as in the Compton case.

To determine the electron energy, generate another random number r_2 in the range $(-1,+1)$. Then, y ($0 \le y \le 1$) is determined from $|r_2|$ as in the Compton case. Adopt $-y$ instead of y if $r_2 \le 0$.

The helicity of electron is given by

$$h_e = \frac{h_L G_{2n} + h_\gamma G_{4n}}{G_{1n} + h_L h_\gamma G_{3n}}. \tag{5.138}$$

The positron momentum is calculated by the momentum conservation and the helicity from the above formula with y replaced by $-y$.

5.9 Beamstrahlung

5.9.1 Basic formulas

When the orbit of a high energy electron(positron) with energy $E_0 = mc^2\gamma$ is bent by a magnetic field B or by an electric field $\mathcal{E}$ with the curvature radius ρ, the critical energy of synchrotron radiation is given by

$$E_c = \overline{\Upsilon} E_0, \qquad \overline{\Upsilon} \equiv \frac{3}{2}\Upsilon, \quad \Upsilon = \gamma^2 \frac{\lambda_C}{\rho} = \gamma \frac{B}{B_{Sch}} = \gamma \frac{\mathcal{E}}{\mathcal{E}_{Sch}} \tag{5.139}$$

where λ_C is the Compton wavelength, $B_{Sch} = m^2/e = 4.4\times 10^9$Tesla is Schwinger's critical field and $\mathcal{E}_{Sch} = cB_{Sch} = 1.32 \times 10^{18}$V/m.

The energy spectrum of emitted photons is given by the Sokolov-Ternov formula. (Radiation angle is not included in **CAIN**. All the photons are emitted forward.) The number of photons per unit time in the interval $(x, x{+}dx)$ of the energy fraction $x = E_\gamma/E_0$ is

$$dW = \frac{\alpha m}{\sqrt{3}\pi\gamma}F_{00}dx, \qquad F_{00} = Ki_{5/3}(z) + \frac{x^2}{1-x}K_{2/3}(z). \tag{5.140}$$

Here, α is the fine structure constant and

$$z = \frac{E_\gamma}{E_c}\frac{1}{1-E_\gamma/E_0} = \frac{E_\gamma E_0}{E_c E'} = \frac{1}{\overline{\Upsilon}}\frac{E_\gamma}{E'} = \frac{1}{\overline{\Upsilon}}\frac{x}{1-x}. \tag{5.141}$$

$E' = E_0 - E_\gamma = E_0(1-x)$ is the final electron energy, K_ν the modified Bessel function and Ki_ν is its integral:

$$Ki_\nu(z) = \int_z^\infty K_\nu(z)dz. \tag{5.142}$$

The function F_00 is available as a **CAIN** function `FuncBS`, and so are some of the K_ν's and Ki_ν's. (See Sec.2.6).

5.9.2 Algorithm of event generation

The random number generation using the acception-rejection method is applicable when the distribution function is everywhere finite and is most efficient when the function is flat.

Since the function F_{00} is infinite at $E_\gamma \to 0$, the following variable y is introduced in **CAIN** instead of the photon energy fraction x in order to make the distribution function finite and relatively flat.[5]

$$x = \frac{\overline{\Upsilon}y^3}{1-y^3+\frac{1}{2}\overline{\Upsilon}(1+y^6)}, \qquad (0<y<1), \tag{5.143}$$

The number of photons during a time interval Δt in the photon energy range $(y, y{+}dy)$ is then given by

$$dn_\gamma = p_0 G(\Upsilon, y)dy, \tag{5.144}$$

where

$$p_0 = \frac{c_0}{\sqrt{3}\pi}\frac{1}{(1+\frac{1}{2}\overline{\Upsilon})^{1/3}}\frac{\alpha\gamma\Delta t}{\rho}, \qquad \left(\frac{\gamma\Delta t}{\rho} = \frac{\Delta t}{\lambda_C}\frac{\mathcal{E}}{\mathcal{E}_{Sch}}\right) \tag{5.145}$$

$$G(\Upsilon, y) = \frac{(1+\frac{1}{2}\overline{\Upsilon})^{1/3}}{c_0\Upsilon}\frac{dx}{dy}F_{00}, \qquad c_0 = \frac{9}{2^{1/3}}\Gamma(2/3). \tag{5.146}$$

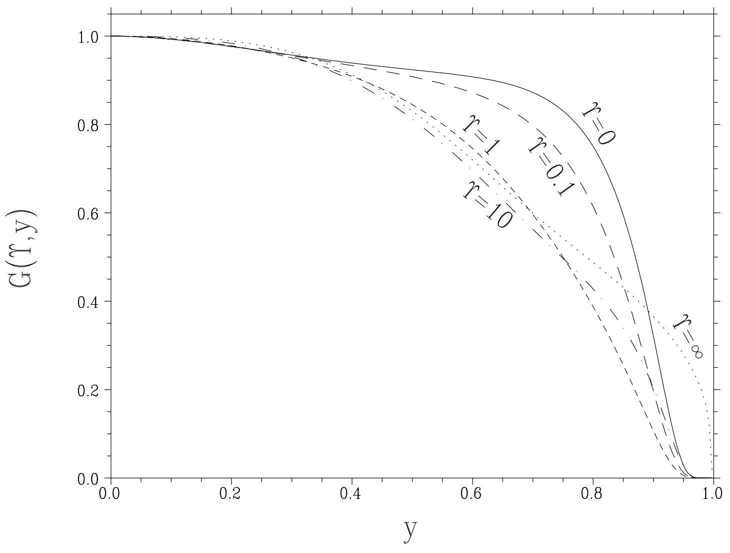


Figure 5.8: Function $G(\Upsilon, y)$ for various values of Υ. Unpolarized case only.

The function $G(\Upsilon, y)$ is less than or equal to unity for any Υ and y. It is plotted in Fig.5.8.

The photon generation in **CAIN** proceeds in the following way.

(1) Calculate p_0[6] for given field, electron energy, and time interval Δt.

(2) Generate one random number p which is uniform in (0,1).

(3) If $p \geq p_0$, reject emitting a photon. Otherwise,

(4) Generate one more random number y uniform in (0,1).

(5) Calculate $G(\Upsilon, y)$. (A polynomial approximation is used for $K_{2/3}$ and $Ki_{5/3}$. The relative error is less than 10^{-4}.)

(6) If $p \geq p_0 G(\Upsilon, y)$, reject emitting a photon. Otherwise,

(7) Emit a photon whose energy is given by eq.(5.143).

The cases when accepted in (3) but rejected in (6) cause a waste of time because the calculation of $G(\Upsilon, y)$ is the most time consuming. The probability to be accepted in (6) is plotted in Fig.5.9 is given by $\int_0^1 G(\Upsilon, y)dy$ and is plotted as a function of Υ. One finds the probability is very high for any Υ owing to the choice of the variable y.

[5] The definition of y has changed since **ABEL** and **CAIN**2.3 in order to keep high efficiency even when Υ is extremely large.

[6] This must be much smaller than 1. Otherwise, the probability of emitting more than one photon during Δt cannot be ignored. The maximum p_0 is defined by the keyword PMAX. When Υ is very large, p_0 is suppressed by the factor $(1 + \frac{1}{2}\overline{\Upsilon})^{1/3}$. However, in defining Δt (by the number of steps in PUSH command) so as to make p_0 small enough, you have to omit this factor because the energy of some electrons can be much smaller after first radiation.

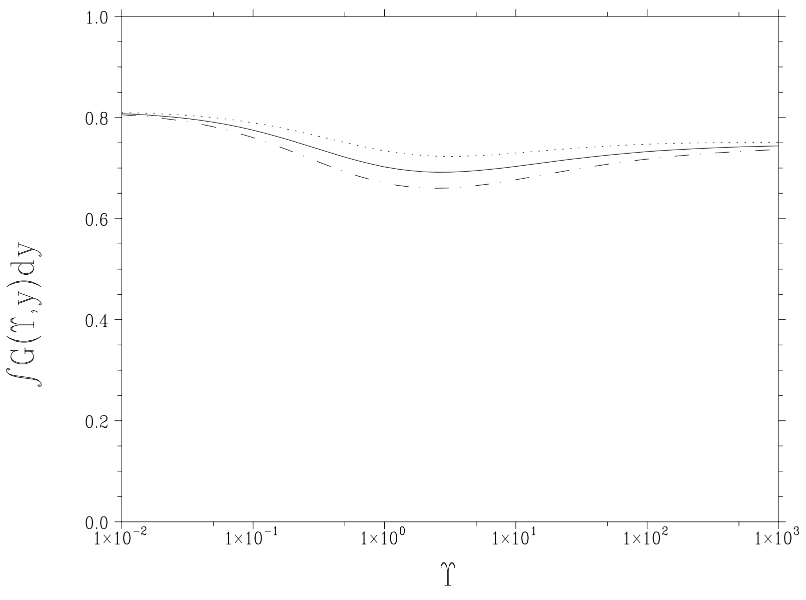


Figure 5.9: The acception probability in the step (6) as a function of Υ. The solid line is the unpolarized case The dot-dash and dotted lines are polarized cases with $\boldsymbol{\zeta}_i\cdot\boldsymbol{e}_2 = 1$ and -1, respectively.

5.9.3 Polarization

Polarization effects have been added since **CAIN**2.1. In order to describe polarizations, let us introduce an orthonormal basis vector ($\boldsymbol{e}_1$, $\boldsymbol{e}_2$, $\boldsymbol{e}_v$). Here, $\boldsymbol{e}_v$ is the unit vector along the initial electron velocity, $\boldsymbol{e}_1$ the unit vector along the direction of the transverse component of acceleration, and $\boldsymbol{e}_2 = \boldsymbol{e}_v\times\boldsymbol{e}_1$. If the acceleration is due to a transverse magnetic field, $-\boldsymbol{e}_2$ is the unit vector along the magnetic field (times the sign of charge). The transition rate from the initial electron polarization $\boldsymbol{\zeta}_i$ to the final polarization $\boldsymbol{\zeta}_f$ with the photon Stokes parameter $\boldsymbol{\xi}$ (based on the basis vector ($\boldsymbol{e}_1$, $\boldsymbol{e}_2$, $\boldsymbol{e}_v$)) is

$$\begin{aligned} dW = \frac{\alpha dE_\gamma}{4\sqrt{3}\pi\gamma^2} \times \Biggl\{ & F_{00}(1+\boldsymbol{\zeta}_i\cdot\overline{\boldsymbol{\zeta}}_f) - xK_{1/3}(\boldsymbol{e}_2\cdot\boldsymbol{\zeta}_i) - \frac{x}{1-x}K_{1/3}(\boldsymbol{e}_2\cdot\overline{\boldsymbol{\zeta}}_f) \\ & -\frac{x^2}{1-x}\left[(\boldsymbol{e}_v\cdot\boldsymbol{\zeta}_i)(\boldsymbol{e}_v\cdot\overline{\boldsymbol{\zeta}}_f)Ki_{1/3} + (\boldsymbol{\zeta}_i\cdot\overline{\boldsymbol{\zeta}}_f - \boldsymbol{e}_v\cdot\boldsymbol{\zeta}_i\boldsymbol{e}_v\cdot\overline{\boldsymbol{\zeta}}_f)K_{2/3}\right] \\ & +\frac{x}{1-x}K_{1/3}(\boldsymbol{e}_1\cdot\boldsymbol{\zeta}_i)\overline{\xi}_1 - \frac{x(2-x)}{2(1-x)}K_{2/3}(\boldsymbol{e}_v\cdot\boldsymbol{\zeta}_i)\overline{\xi_2} + \left[K_{2/3} - \frac{x}{1-x}K_{1/3}(\boldsymbol{e}_2\cdot\boldsymbol{\zeta}_i)\right]\overline{\xi_3}\Biggr\} \end{aligned} \tag{5.147}$$

where F_{00} is defined in eq.(5.140) and the argument of the Bessel functions is z. We omitted the terms involving $\boldsymbol{\zeta}_f$ and $\boldsymbol{\xi}$ simultaneously, which means to ignore the correlation of polarization between the final electron and photon. (See Sec.5.2.2 for the meaning of bars on $\boldsymbol{\zeta}_f$ and $\boldsymbol{\xi}$.)

The radiation energy spectrum summed over the final polarization is given by eq.(5.140) with F_{00} replaced by

$$F_0 = F_{00} - xK_{1/3}(\boldsymbol{e}_2\cdot\boldsymbol{\zeta}_i). \tag{5.148}$$

Since the function $G(\Upsilon, y)$ with F_{00} replaced by F_0 has still the above mentioned property, the same algorithm of generating the photon energy can be used. ($G(\Upsilon, y)$ is slightly larger when $\boldsymbol{e}_2\cdot\boldsymbol{\zeta}_i = -1$ but still $G(\Upsilon, y) \leq 1$.)

For the given radiation energy $E_\gamma = xE_0$, the polarizations of the final electron and photon are calculated by the prescription described in Sec.5.2.2. Thus,

$$\boldsymbol{\zeta}_f = \frac{1}{F_0}\left\{F_{00}\boldsymbol{\zeta}_i - \frac{x}{1-x}K_{1/3}\boldsymbol{e}_2 - \frac{x^2}{1-x}\left[(\boldsymbol{e}_v\cdot\boldsymbol{\zeta}_i)\boldsymbol{e}_v Ki_{1/3} + [\boldsymbol{\zeta}_i - (\boldsymbol{e}_v\cdot\boldsymbol{\zeta}_i)\boldsymbol{e}_v]K_{2/3}\right]\right\} \quad (5.149)$$

$$\xi_1 = \frac{x}{1-x}\frac{K_{1/3}}{F_0}(\boldsymbol{e}_1\cdot\boldsymbol{\zeta}_i), \quad \xi_2 = -\frac{x(2-x)}{2(1-x)}\frac{K_{2/3}}{F_0}(\boldsymbol{e}_v\cdot\boldsymbol{\zeta}_i), \quad \xi_3 = \frac{K_{2/3} - \frac{x}{1-x}K_{1/3}(\boldsymbol{e}_2\cdot\boldsymbol{\zeta}_i)}{F_0} \quad (5.150)$$

In the case when the event generation is rejected, the polarization of the electron must be changed according to eq.(5.12):

$$\boldsymbol{\zeta}_f = \boldsymbol{\zeta}_i + [\boldsymbol{e}_2 - (\boldsymbol{e}_2\cdot\boldsymbol{\zeta}_i)\boldsymbol{\zeta}_i]\int_0^1 \frac{\alpha m}{\sqrt{3}\pi\gamma}xK_{1/3}(z)dx. \quad (5.151)$$

In storage rings, the electron polarization builds up slowly along the direction of the magnetic field. This effect comes from the difference between the coefficient of $\boldsymbol{\zeta}_i$ and $\overline{\boldsymbol{\zeta}}_f$ (see eq.(5.14)):

$$\frac{x}{1-x}K_{1/3} - xK_{1/3} = \frac{x^2}{1-x}K_{1/3}. \quad (5.152)$$

When Υ is small ($\lesssim 10^{-5}$ in storage rings), each term on the left hand side is proportional to Υ whereas the right hand side is Υ^2 because of cancellation. **CAIN** cannot reproduce such slow buildup, even if the computing time allows, because the approximate polynomials adopted do not have that accuracy. They are enough, however, for beam-beam problems.

5.9.4 Enhancement factor of the event rate

CAIN normally produces macro-photons such that the expected number of macro-photons per macro-electron is equal to the expected number of real photons per real electron. In some cases, however, too many macro-photons are created causing the memory overflow, or the statistics is too poor due to a small number of macro-photons. To solve this problem, a variable `WENHANCEMENT`$=w_{enh}$ is introduced in the `CFQED` command.

When $w_{enh} > 1$, more macro-photons are created. They have the weight smaller than that of the parent electron/positron by the factor $1/w_{enh}$. However, the recoil of electron/positron is taken into account only with the probability $1/w_{enh}$ so that their statistical property does not depend on w_{enh}.

When $w_{enh} < 1$, the event generation goes the same as in the case w_{enh}=1, but the final photons are stored in the memory only with the probability w_{enh}. The recoil of electron/positron is taken into acount regardless the photon is stored or not.

Thus, if there is no bug, w_{enh} does not cause any physical change.

5.10 Coherent Pair Creation

5.10.1 Basic formulas

When a high energy photon goes through a strong transverse field, it can create a real electron-positron pair. This process is known as 'coherent pair creation' and is characterized by the parameter

$$\chi = \frac{E_\gamma}{mc^2}\frac{B}{B_{Sch}} = \frac{E_\gamma}{mc^2}\frac{\mathcal{E}}{\mathcal{E}_{Sch}}, \tag{5.153}$$

where E_γ is the energy of the initial photon. (B_{Sch} and $\mathcal{E}_{Sch}$ are defined in eq.(5.139).) The probability of the process is exponentially small ($\propto e^{-8/(3\chi)}$) when χ is small.

Let us denote the energy and polarization vector of initial photon and final positron/ electron by E_γ, $E_\pm$ ($E_+ + E_- = E_\gamma$), $\boldsymbol{\xi}$, and $\boldsymbol{\zeta}_\pm$. The transition rate is obtained by the following replacement in the formula (5.147):

$$\begin{aligned}
&E_\gamma \to -E_\gamma, \qquad E_0 \to -E_+, \qquad E' \to E_-, \qquad E_\gamma^2 dE_\gamma \to -E_+^2 dE_+,\\
&(\overline{\xi}_1, \overline{\xi}_2, \overline{\xi}_3) \to (\xi_1, -\xi_2, \xi_3),\\
&z = \frac{2}{3\Upsilon}\frac{E_\gamma}{E'} \to \eta = \frac{2}{3\chi}\frac{E_\gamma^2}{E_+E_-}.
\end{aligned} \tag{5.154}$$

Ignoring the terms related to the polarization correlation between the final electron and positron, we obtain

$$\begin{aligned}
dW = \frac{\alpha m^2 dE_+}{4\sqrt{3}\pi E_\gamma^2} \times \Bigg\{ & F_{CP} - K_{2/3}\xi_3 + \frac{E_\gamma(E_+ - E_-)}{2E_+E_-}K_{2/3}\boldsymbol{e}_n\cdot(\overline{\boldsymbol{\zeta}}_+ - \overline{\boldsymbol{\zeta}}_-)\xi_2 \\
& + K_{1/3}\boldsymbol{e}_2\cdot\left(\frac{E_\gamma}{E_+}\overline{\boldsymbol{\zeta}}_+ + \frac{E_\gamma}{E_-}\overline{\boldsymbol{\zeta}}_-\right) + K_{1/3}(\boldsymbol{e}_1\xi_1 + \boldsymbol{e}_2\xi_3)\cdot\left(\frac{E_\gamma}{E_-}\overline{\boldsymbol{\zeta}}_+ + \frac{E_\gamma}{E_+}\overline{\boldsymbol{\zeta}}_-\right)\Bigg\}
\end{aligned} \tag{5.155}$$

where F_{CP} is defined as

$$F_{CP} = Ki_{1/3} + \left(\frac{E_-}{E_+} + \frac{E_+}{E_-}\right)K_{2/3}. \tag{5.156}$$

K_ν is the modified Bessel function and Ki_ν is defined in eq.(5.142). Their arguments are η defined in eq.(5.154).

The transition rate summed over the final polarization is

$$dW = \frac{\alpha m^2 dE_+}{\sqrt{3}\pi E_\gamma^2}(F_{CP} - K_{2/3}\xi_3). \tag{5.157}$$

The function $F_{CP} - K_{2/3}\xi_3$ is plotted in Fig.5.10 as a function of E_+/E_γ. The function F_{CP} is available as a **CAIN** function `FuncCP`, and its integral over $0 < E_+/E_\gamma < 1$ as `IntFCP`. (See Sec.2.6).

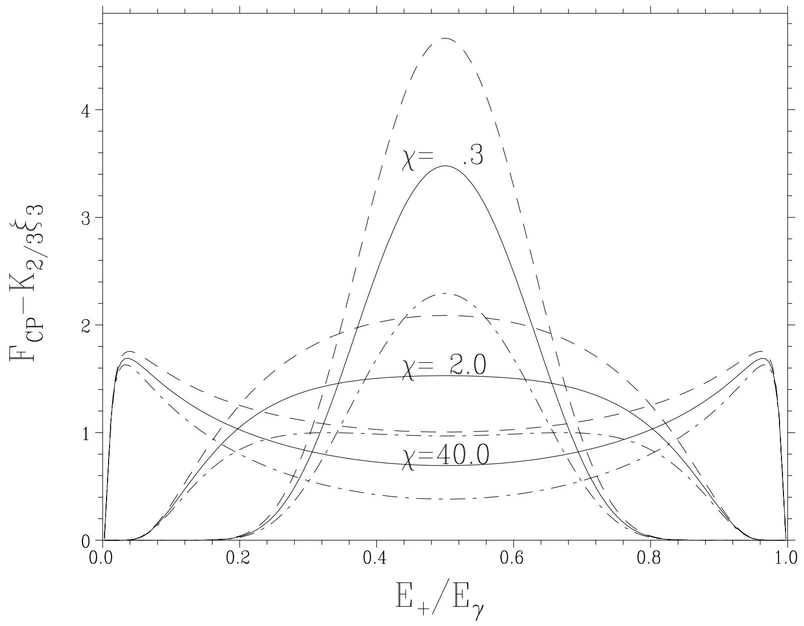


Figure 5.10: Function $F_{CP} - K_{2/3}\xi_3$ for three values of χ. The solid, dot-dash and dashed curves are for $\xi_3 = 0,\ 1,\ -1$, respectively. The curves for $\xi_3 = 0$ are normalized such that $\int F_{CP} dE_+/E_\gamma = 1$, and those for $\xi_3 \neq 0$ are drawn with the same scale as the corresponding $\xi_3 = 0$ curves.

5.10.2 Algorithm of event generation

The total rate for given $\boldsymbol{\xi}$ is approximately

$$W \approx W_{app} \equiv \frac{\alpha m^2}{E_\gamma} U, \qquad U(\chi, \xi_3) \equiv e^{-\eta_0} \left\{ \frac{\chi}{[c_1^3 + c_2^3 \chi]^{1/3}} - \frac{\xi_3 \chi}{[(3c_1)^3 + (5c_2)^3 \chi]^{1/3}} \right\}, \tag{5.158}$$

where

$$\eta_0 \equiv \frac{8}{3\chi}, \qquad c_1 = \frac{16\sqrt{2}}{3\sqrt{3}}, \qquad c_2 = \frac{7\pi}{5} \frac{2^{2/3}}{3^{2/3}} \frac{1}{[\Gamma(5/6)]^2}.$$

In order to make the spectrum function flatter, **CAIN** introduces the variable y ($-1 < y < 1$) instead of $x = E_+/E_\gamma$ ($0 < x < 1$):

$$x = \frac{1}{2}\left(1 + \mathrm{sgn}\, y \sqrt{\frac{y'}{\eta_0 + y'}}\right), \qquad y' = -\log\left\{1 - \frac{y^2}{[1 + (1-y^2)/(2\sqrt{\eta_0})]^2}\right\}. \tag{5.159}$$

The spectrum function with respect to y then becomes

$$\frac{dW}{d(y/2)} = \frac{\alpha m^2}{E_\gamma} c_3 U(\chi, \xi_3) \tilde{F}(y, \chi, \xi_3), \qquad \tilde{F} \equiv \frac{(F_{CP} - K_{2/3}\xi_3)/\sqrt{3}\pi}{c_3 U} \frac{dx}{d(y/2)}, \tag{5.160}$$

where $y/2$ is used because the range of y is 2. The constant

$$c_3 = \frac{7\sqrt{\pi}}{6\Gamma(5/6)\Gamma(2/3)} = 1.35286\ldots.$$

is chosen so that $\tilde{F} \leq 1$ for any (y, χ, ξ_3). $\tilde{F}(y, \chi, \xi_3)$ is plotted in Fig.5.11.

Now, the event generation goes as

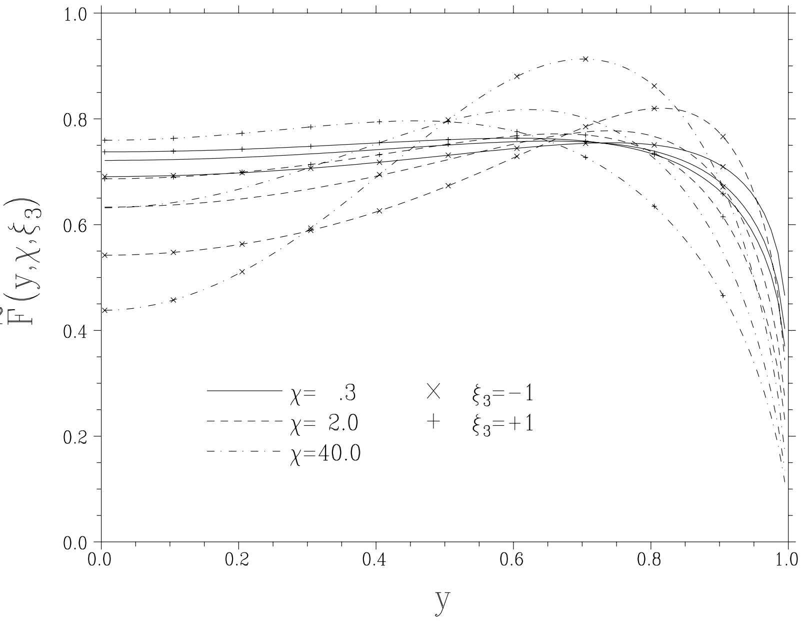


Figure 5.11: Function $\tilde{F}(y,\chi,\xi_3)$ as a function of y for three values of χ =0.3, 2, 40 and for $\xi_3 = 0,\pm1$. The parameter χ is indicated by the line mode and ξ_3 is by crosses (no cross for $\xi_3 = 0$).

(1) Compute χ and reject if $\chi < 0.05$ (the rate is too small).

(2) Generate a uniform random number $0 < p < 1$ and compute $p_0 \equiv c_3 W_{app}\Delta t$.

(3) Reject if $p > p_0$.

(4) Generate another uniform random numbers $0 < q < 1$ and $-1 < y < 1$ and compute $\tilde{F}$. (Instead of q, one can also use p/p_0.)

(5) Reject if $q > \tilde{F}$.

(6) Accept and compute x from eq.(5.159) and $E_+ = xE_\gamma$, $E_- = (1-x)E_\gamma$.

(7) Compute the final polarization of $e^\pm$ from

$$(F_{CP} - K_{2/3}\xi_3)\boldsymbol{\zeta}_+ = \frac{1}{x}K_{1/3}\boldsymbol{e}_2 + \frac{1}{1-x}K_{1/3}(\xi_1\boldsymbol{e}_1 + \xi_3\boldsymbol{e}_2) + \frac{2x-1}{2x(1-x)}K_{2/3}\xi_2\boldsymbol{e}_n \tag{5.161}$$

The formula for $\boldsymbol{\zeta}_-$ is obtained by replacing x by $1-x$.

(8) In the case of rejection, the polarization of the photon should be changed according to eq.(5.12):

$$\xi_{1,fin} = \xi_1 - a\xi_3\xi_1, \quad \xi_{2,fin} = \xi_2 - a\xi_3\xi_2, \quad \xi_{3,fin} = \xi_3 + a(1-\xi_3^2), \tag{5.162}$$

$$a = \frac{\alpha m^2 \Delta t}{\sqrt{3}\pi E_\gamma}\int_0^1 K_{2/3}\left(\frac{2}{3\chi}\frac{1}{x(1-x)}\right)dx.$$

The error of the formula (5.158) does not cause any inaccuracy of event generation (but causes inefficiency).

5.11 Incoherent Processes

In addition to the interactions between a particle and a macro-scopic field such as beamstrahlung and laser-Compton, there are particle-particle interactions between e$^-$, e$^+$ and γ that have to be simulated. The present version of **CAIN** include the following processes:

Breit-Wheeler	$\gamma + \gamma \to \mathrm{e}^- + \mathrm{e}^+$
Bethe-Heitler	$\gamma + \mathrm{e}^\pm \to \mathrm{e}^\pm + \mathrm{e}^- + \mathrm{e}^+$
Landau-Lifshitz	$\mathrm{e} + \mathrm{e} \to \mathrm{e} + \mathrm{e} + \mathrm{e}^- + \mathrm{e}^+$
Bremsstrahlung	$\mathrm{e} + \mathrm{e} \to \mathrm{e} + \mathrm{e} + \gamma$

These QED processes have relatively large event rates even at high energies.

The treatment of these incoherent processes in **CAIN** is slightly different from other (coherent) processes since the event rates of the former are usually much smaller than the latter.

- The parent macro particles are not eliminated nor changed after interaction. The polarization change described in Sec.5.2.2 is not taken into account.
- More than one event may be generated in one time step because the change of parent partciles after the first event need not be taken into account.

In contrast to **ABEL**, **CAIN** does not assume the particles are ultra-relativistic. Therefore, unless the beam-beam field created by the pair particles becomes significant, their trajectories are correctly calculated. (Note, however, that only the beam field due to the on-coming beam is taken into account in the present version. The beam field in the same beam may have significant effects on low energy particles.)

Among the four QED processes above, the Breit-Wheeler process is treated as a fundamental process. Others are reduced to the former (or to the Compton process, in the case of Bremsstrahlung) by using the virtual photon approximation. As for the polarization, only the circular polarization of initial photons in the direct (non-virtual) Breit-Wheeler process is taken into account.

5.11.1 Breit-Wheeler Process

The differential crosssection with respect to the scattering angle of the final electron in the center-of-mass frame is given by

$$\frac{d\sigma_{BW}}{dc} = \frac{\pi}{2} r_e^2 \frac{m^2}{\omega^2} f, \qquad f = \frac{p}{\omega}(f_0 - f_2 h) \tag{5.163}$$

with

$$f_0 = \frac{\omega^2 + p^2c^2}{\omega^2 - p^2c^2} + \frac{1}{2}\left[1 - \left(1 - \frac{2m^2}{\omega^2 - p^2c^2}\right)^2\right] \tag{5.164}$$

$$f_2 = \frac{\omega^2 + p^2c^2}{\omega^2 - p^2c^2}\left(1 - \frac{2m^2}{\omega^2 - p^2c^2}\right) \tag{5.165}$$

where

ω, p	Energy and momentum of final electron in the center-of-mass frame.
c	Cosine of the scattering angle θ of the final electron in the center-of-mass frame.
h	Product of circular polarizations of the two initial photons.

The total crosssection is

$$\sigma_{BW} = \pi r_e^2 \frac{m^2}{\omega^2} G \tag{5.166}$$

where

$$G = \int_0^1 f\,dc = 2\left(1 - h + \frac{2a^2 - 1}{2a^4}\right)\sinh^{-1} b + \frac{b}{a}\left[-(1 + \frac{1}{a^2}) + 3h\right], \tag{5.167}$$

where

$$a = \omega/m, \qquad b = p/m, \qquad a^2 = b^2 + 1.$$

Events are generated by the following algorithm using inverse function.

(a) Compute ω, p, a, b, G and σ for given initial parameters (reject if $a \le 1$, *i.e.*, below threshold) and calculate the event probability for the given time step

$$P = \frac{w_1 w_2}{w} \frac{\sigma_{BW} \Delta t}{V}$$

where w_1 and w_2 are the weights of initial photons (number of real photons divided by that of macro photons), w the weight of the pair to be created, Δt the time interval and V the volume in which the macro phtons are located.

(b) If P is too large (say, > 0.1), divide the interval Δt (and P) by an integer n_{div}, and repeat the following procedure n_{div} times.

(c) Generate a random number $r_1 \in (0, 1)$. Reject if $r_1 \ge P$.

(d) Generate another random number $r_2 \in (-1, 1)$ and solve the equation

$$2\left(1 - h + \frac{2a^2 - 1}{2a^4}\right) z - (1 - h)\tanh z + \frac{2a^2 h - 1}{a^4} \sinh z \cosh z = |r_2|\, G \tag{5.168}$$

with respect to z. Here z is defined by $|c| = |\cos\theta| = (a/b)\tanh z$ $(0 \le z \le \sinh^{-1} b)$. The left hand side is the integral of f from 0 to $|c|$. The sign of $c = \cos\theta$ is determined by the sign of r_2.

(e) Generate another random number $r_3 \in (0, 2\pi)$ and compute the transverse component of electron momentum by

$$\boldsymbol{p}_\perp = p \sin\theta(\boldsymbol{e}_1 \cos\phi + \boldsymbol{e}_2 \sin\phi) \tag{5.169}$$

where $\boldsymbol{e}_1$ and $\boldsymbol{e}_2$ are arbitrary unit vectors perpendicular to $\boldsymbol{e}_3$, the unit vector along the initial photon momentum in the center-of-mass frame. The latter is given by

$$\boldsymbol{e}_3 = \frac{(\omega+\omega_2)\boldsymbol{k}_1 - (\omega+\omega_1)\boldsymbol{k}_2}{\omega(2\omega+\omega_1+\omega_2)}$$

where $(\omega_j, \boldsymbol{k}_j)$ are the energy momentum of the photons in the original frame. The value of $\sin\theta$ should be computed from

$$\sin^2\theta = \frac{b^2 - \sinh^2 z}{b^2\cosh^2 z}, \qquad (\sin\theta \geq 0)$$

rather than from $|\cos\theta|$ because the latter is usually very close to unity when ω is much larger than the electron rest mass.

(f) Then, the momentum of the electron in the original frame is calculated by

$$\boldsymbol{p} = \frac{1}{2}\left[(1+\frac{pc}{\omega})\boldsymbol{k}_1 + (1-\frac{pc}{\omega})\boldsymbol{k}_2\right] + \boldsymbol{p}_\perp + \frac{\boldsymbol{k}\cdot\boldsymbol{p}_\perp}{2\omega+\omega_1+\omega_2}\frac{\boldsymbol{k}}{2\omega} \qquad (5.170)$$

where $\boldsymbol{k} = \boldsymbol{k}_1 + \boldsymbol{k}_2$. Note that $1 - p\,|c|\,/\omega$ must be computed from $e^{-z}/\cosh z$ in order to avoid round off errors.

(g) The momentum of positron is computed from the momentum conservation.

5.11.2 Virtual (almost real) photon approximation

To treat the Bethe-Heitler, Landau-Lifshitz and the Bremsstrahlung processes, the so-called almost-real-photon approximation, or equivalent photon approximation, or Weizäcker-Williams approximation, is employed. An electron is accompanied by virtual photons which look like real photons at ultra-relativistic limit. They interact with on-coming (real or virtual) photons incoherently. Thus, the Bethe-Heitler and Landau-Lifshitz processes above are reduced to the Breit-Wheeler process and the Bremsstrahlung to the Compton process:

Bethe-Heitler	$\gamma + `\gamma' \to \mathrm{e}^- + \mathrm{e}^+$
Landau-Lifshitz	$`\gamma' + `\gamma' \to \mathrm{e}^- + \mathrm{e}^+$
Bremsstrahlung	$\mathrm{e} + `\gamma' \to \mathrm{e} + \gamma$

where 'γ' is a virtual photon.

Let the electron energy be $E = m\gamma$ ($\gamma \gg 1$). The number of virtual photons with energy ω and transverse momentum $\boldsymbol{q}_\perp$ is given by

$$dn = \frac{\alpha}{\pi}\frac{d\omega}{\omega}\,\frac{1}{\pi}\frac{\boldsymbol{q}_\perp^2 d\boldsymbol{q}_\perp}{(\boldsymbol{q}_\perp^2 + \omega^2/\gamma^2)^2}, \qquad (|\boldsymbol{q}_\perp| \lesssim m) \qquad (5.171)$$

where α is the fine structure constant. For given ω, the typical transverse momentum is very small, $q_\perp \sim \omega/\gamma$, so that it is not important in collision kinematics but, instead, the finite transverse extent $\sim 1/q_\perp$ can bring about significant effects. In the (transverse) configuration space, the above expression becomes

$$dn = \frac{\alpha}{\pi}\frac{d\omega}{\omega}\,K_1^2(\frac{\omega\rho}{\gamma})\frac{\omega}{\gamma^2}\frac{d\boldsymbol{r}_\perp}{\pi}, \qquad (\rho \gtrsim 1/m) \qquad (5.172)$$

where $\boldsymbol{r}_\perp$ is the transverse coordinate with respect to the parent electron, $\rho = |\boldsymbol{r}_\perp|$, and K_1 the modified Bessel function.

The transverse momentum cut off $|\boldsymbol{q}_\perp| \lesssim m$ (or $\rho \gtrsim 1/m$) is somewhat umbiguous. It should depend on the momentum transfer of the whole process. This dependence is ignored in **CAIN** because the virtual photons are generated independently from the following processes and because it does not much affect the low energy pairs.

The lower limit ω_{min} of the integration over ω is, in our case, determined by the pair creation threshold. Let us introduce dimensionless variables $y = \omega/E$, $y_{min} = \omega_{min}/E$, and $x = \omega\rho/\gamma$. The total number of the virtual photons is given by[7]

$$n = \frac{\alpha}{\pi}\int_{y_{min}}^{1}\frac{dy}{y}\int_{y}^{\infty}K_1^2(x)2xdx \tag{5.173}$$

$$= \frac{\alpha}{\pi}\int_{y_{min}}^{1}\frac{dy}{y}V(y), \tag{5.174}$$

with

$$V(x) = x^2[K_0(x)K_2(x) - K_1^2(x)]. \tag{5.175}$$

When $y_{min} \ll 1$, the total number is

$$n = \frac{\alpha}{\pi}\left[\log^2 y_{min} - (2\log 2 - 2\gamma_{\scriptscriptstyle E} - 1)\log y_{min}\right], \tag{5.176}$$

where $\gamma_{\scriptscriptstyle E} = 0.577\ldots$ is Euler's constant. At very high energies the number of virtual photons per electron is $O(1)$, in spite of the small factor α/π, due to the factor $\log^2 y_{min}$.

5.11.3 Numerical methods

When Bethe-Heitler and/or Landau-Lifshitz processes are specified by `PPINT` command, **CAIN** generates virtual photons in each longitudinal slice at each time step and counts them in the same mesh as that generated by the `LUMINOSITY` command. The number of macro-virtual photons is somewhat arbitrary. In the present version it is determined such that the weight of the macro-virtual photons is equal to the maximum weight of the electrons in the on-coming beam (not equal to the weight of each parent electron in order to prevent low-weight electrons from generating many photons).

Since the y (energy) spectrum is approximately proportional to $\log y/y$ for small y, the spectrum becomes almost flat if one chooses $\log^2 y$ as the primary variable. To account for relatively large y too, **CAIN** adopts the variable η instead of y:

$$y = \exp\left[-\frac{\eta}{\sqrt{c+\eta}}\right], \qquad \eta = \frac{1}{2}\left(\log^2 y + \sqrt{\log^4 y + 4c\log^2 y}\right). \tag{5.177}$$

Here, $c > 0$ is introduced so that the function $G(\eta)$ defined later, is finite. It is chosen to be 0.2 but is almost arbitrary provided $c \gtrsim 0.1$. The maximum η is

$$\eta_{max} = \frac{1}{2}\left(\log^2 y_{min} + \sqrt{\log^4 y_{min} + 4c\log^2 y_{min}}\right). \tag{5.178}$$

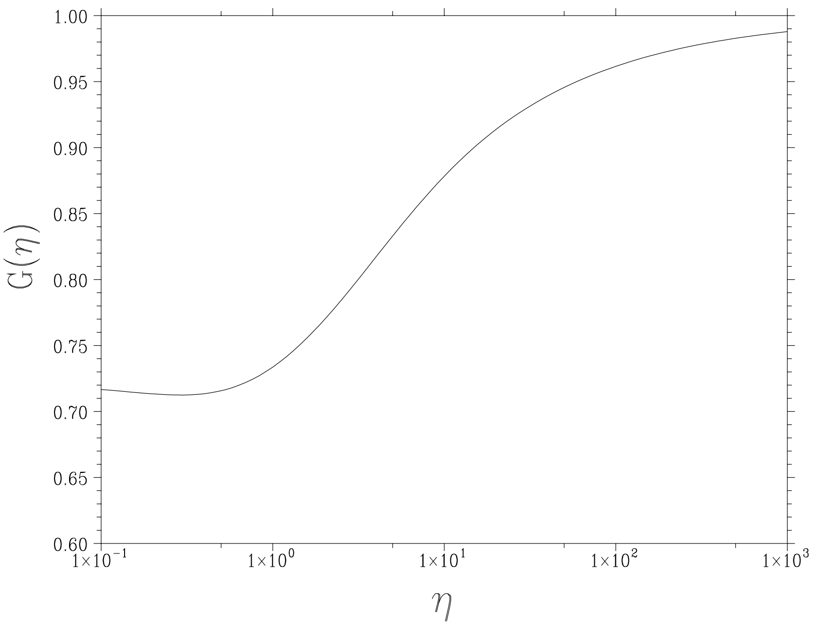


Figure 5.12: Function $G(\eta)$ defined in eq.(5.180). It is close to unity because only large η region is important. $G(0)$ is finite and depends on the parameter c. $G(0) < 1$ if $c > 0.1035\ldots$. Here, $c = 0.2$ is adopted.

Now, the spectrum with respect to η is

$$dn = \frac{\alpha}{\pi} G(\eta) d\eta, \tag{5.179}$$

with

$$G(\eta) = \frac{2c + \eta}{2(c + \eta)^{3/2}} V(y). \tag{5.180}$$

For $0 < \eta < \infty$, $G(\eta) \leq 1$ and close to 1 except for the small η region which is umimportant in practice. Thus,

$$n = \frac{\alpha}{\pi} \int_0^{\eta_{max}} G(\eta) d\eta \leq \frac{\alpha}{\pi} \eta_{max}. \tag{5.181}$$

For given η (or y) the distribution of x is proportional to $dV(x)/V(y)$ and, therefore, can be random-generated by using inverse function $V^{-1}(V)$.

The algorithm is as follows.

(a) From the given parameters, compute y_{min}, η_{max} and

$$P_0 \equiv \frac{\alpha}{\pi} \frac{\eta_{max}}{w}$$

where w is the weight of virtual photon to be created. P_0 is the expected number of macro-virtual photons. If P_0 is not small enough (say, > 0.1), divide it by an integer N and repeat the following steps N times.

[7] The upper limit $y = 1$ is not regorous. The recoil effect must be taken into account when y is large.

(b) Generate a uniform random number $r_1 \in (0, 1)$. Reject if $r_1 \geq P_0$. Otherwise redefine r_1 by r_1/P_0.

(c) Generate a random number $r_2 \in (0, 1)$, define $\eta = r_2 \eta_{max}$ and calculate $G(\eta)$ from a table. Reject if $r_1 \geq G$. (The probability to be rejected here is small because G is close to unity.) Otherwise, accept.

(d) Calculate y using eq.(5.177) and $\omega = Ey$. If `LOCAL` option is specified, stop here and return $\boldsymbol{r}_\perp = (0, 0)$. Otherwise, calculate the value of $V(y)$ from G using eq.(5.180).

(e) Generate a random number $r_3 \in (0, 1)$ and solve the equation $r_3 = V(x)/V(y)$ with respect to x. This is done by using a table of inverse function of V.

(f) Compute $\rho = \lambda_e x/y$, λ_e being the Compton wave length.

(g) Generate a random number $r_4 \in (0, 1)$ and compute the photon coordinate $\boldsymbol{r}_\perp = (\rho \cos 2\pi r_4, \rho \sin 2\pi r_4)$.

Appendix A

History of Revision

There can be lots of items (in particular bug-fixes) missing here.

A.1 CAIN2.35

- `PLOT BLGEOMETRY` added. (Apr.22.2002)
- Added `MBBXY` and `MLUMMESH` in `ALLOC` command. Also a bug in the file `rdalloc.f` that `MMAG` and `MBEALINE` in `ALLOC` command had not been recognized was fixed. (Nov.20.2002)
- Bug fix in `vphbfl.f` (Feb.05.2003) (Thanks to K. Moenig)
  `WRITE(TDFL,460) NELEC,WGTESUM,NPH,WGTSUM`
  replaced by
  `WRITE(TDFL,460) NELEC,NPH,WGTESUM,WGTSUM`
- `STORE` command changed so that the variable `MsgFile` is not stored and, therefore, not restored by the `RESTORE` command. This is to avoid a bug on Windows platform when simultaneous output to a file and to the console is intended. (Apr.10.2003)
- Linear laser polarization added in nonlinear Compton scattering. (Apr.21.2003) However, the electron spin is not yet included.
- Files `jobdat.f` and `clock1.f` changed. They now only use standard FORTRAN routines `DATE_AND_TIME` and `SYSTEM_CLOCK`. These files were moved to the directory `src/` and the directort `src/local/` was removed. The file `windows/second.c` was removed. There is no more C files. (Apr.23.2003)
- The redundunt argument for `RANDCAIN` eliminated. (Apr.23.2003)
- Platform-dependent common blocks (word boundary problem due to the length of FORTRAN pointers) `EVLOADCM` and `LASRCM4` were replaced by `MODULE`s. (Apr.25.2003)

A.2 CAIN2.33

- When the destination of MsgFile in Windows version is a file, its copy can also be shown on the console. See Sec.4.2.4. changed. See Sec.3.31

- A predefined particle variable `$PName` added.
- The treatment of `SELECT` operand in `PLOT` and `CLEAR BEAM` commands.
- `PLOT BLGEOMETRY` added.
- Bug fix of error message in rdall.f.
  `CMD(ICMD(IC))(1:NCCMD(ICMD(IC)))` → `CMD(IC)(1:NCCMD(IC))`
- Bug fix for test particles. `P(3)` → `EP(0:3)` in addtstp.f and also line mode errors in pltstp.f, scat.f.
- Bug fix for virtual photon generation in vphgen.f.
  `P0=FINSTRC/PI*HMAX/WGT` → `P0=FINSTRC/PI*HMAX*WGT`
  This bug is serious. It seems, howerver, `WGT` is unity in most applications upto now. It causes an error in incoherent pair creation when an even enhancement function is used in two step processes (like γ-γ).
- Bug fixes related to character variables.
  In evcmpl0.f,
  `ELSEIF(LCH.GE.C_EXP.AND.LCH.LE.C_VAR) THEN` →
  `ELSEIF(LCH.GE.C_EXP.AND.LCH.LE.C_VAR.OR.LCH.EQ.C_DOLLAR) THEN`
  In evload.f. After
  ```
  CALL EVARRGET(-PL(3,IP),0,0,X(PL(2,IP)),ERR1)
  IF(ERR1.NE.' ') GOTO 950
  ```
  2 lines inserted (note the first line is two long. Must be split into 2 lines):
  ```
  CALL FLCHSET2(GSTR(X(PL(2,IP))%C(1):X(PL(2,IP))%C(2)),X(PL(2,IP)),ERR1)
  IF(ERR1.NE.' ') GOTO 940
  ```
  Also the same 2 lines inserted after
  ```
  CALL EVARRGET(ID,PL(3,IP),IND,X(PL(2,IP)),ERR1)
  IF(ERR1.NE.' ') GOTO 950
  ```
- Bug fix in evarrget.f. Inserted `ERR=' '`
  just after the first `RETURN`.

A.3 CAIN2.32

- A predefined particle variable `Incp` added.
- `BEAM SINGLEPARTICLE` added for adding a particle in the particle list.
- Runge-Kutta integration introduced in drift1.f for more accurate computation of `PUSH` in extenal field (but with no beam-beam).
- Change of the third argument of the beam statistics functions (`AvrX` etc). When the particle selection argument is given, the incoherent particles are included by default. If not given, excluded.
- Destination of `MsgFile` in Windows version can be a file. See Sec.4.2.4.
- Bug fix in evdefarr.f.
  After the line `CALL EVARFREE(ID)`, inserted
  `IF(NARRAY.LT.ID) NARRAY=NARRAY+1`

- Bugfix in lsrgeo.f. (Thanks to K. Dobashi)
  After the line `PD0=0` near the beginning of the file, the line `PD=0` was added.
- Bugfix in drfext.f (Thanks to A. Stahl)
  `CHARG0=CHARG` added just after the line
  `ELSEIF(CHARG.NE.CHARG0) THEN`
- Bugfix in cprnd2.f.
  `EXPETA=EXP(-DETA)` → `EXPETA=EXP(-ETA)`
  `FCP=FCOHP*DXDP/FMAX*CC0/WCOHP0` → `FCP=FCOHP*DXDP/FMAX*CC0/WCOHP1`

A.4 CAIN2.31

- `ELSEIF` added.
- DO loop improved
  - `DO i=`(i_1,i_2,i_3) type added. Due to this change, a comma after `DO REPEAT` and `DO WHILE` is now mandatory.
  - `CYCLE` and `EXIT` added
- Initial check of nesting `DO/IF/PUSH/TRANSPORT` added.
- The variables `$InFileName` and `$InFilePath` added.

A.5 CAIN2.3

- The 'expression' greately improved.
  - Arrays introduced (command `ARRAY`). See Sec.2.5.5. `PRINT ARRAY` added.
  - Logical expression introduced. (See Sec.2.5)
  - Character expression introduced. (see Sec.2.5.6.)
- The command `MATCHING` added.
- A new flexible format defined for reading particle files.

These changes have been done mostly in a backward compatible way, but there are a few non-compatible changes.

- The operands `GEN` and `GENERATION` in some commands were eliminated (syntax like `GEN >=2` does not fit to the new grammer), having been replaced by the much more powerful operand `SELECT` for particle selection. See Sec.3.31.
- Character strings such as `FILE='...'` must be enclosed by apostrophes. (Old versions allowed omitting them if the string did not contain blanck spaces, etc.)
- The last (optional) argument of the beam statistics functions (`SigX` etc) has changed its meaning. See Sec.2.6.

- The use of `''` within a character string to express one `'` is better be avoided. You can now use `" ' "`.

Other minor changes:

- Travelling focus parameter `DALPHADE` has been introduced. See Sec.3.5.
- Bug fix in cohpar.f and cprnd2.f (data initialization). (Thanks to D. Asner)

A.6 CAIN2.23

- The commands `MAGNET`, `BEAMLINE`, `BLOPTICS`, `TRANSPORT`, and `ENDTRANSPORT` are added. `PRINT/PLOT BLOPTICS` and `PRINT MAGNETS/BLGEOMETRY` added accordingly.
- Bug fix in lumprt.f (Factor 10^{-4} in luminosity output.) Thanks to G. Franzoni.

A.7 CAIN2.21

- Bug fix in donut.f ($\boldsymbol{n}$ vector)
- Random number routine changed the name: `rand`$\rightarrow$`randcain` to avoid name confliction in some unix systems.
- `makefile` rewritten for jlccc. (Thanks to I. Sakai)
- `@go` rewritten for jlccc. (Thanks to T. Omori)

A.8 CAIN2.2a

- Donut laser beam by axicon introduced.

A.9 CAIN2.2

There has been a version **CAIN**2.1d but the changes since then are only bug fixes. Here we list up the changes beside bug fixes.

- Unequal mesh of energy for differential luminosity has been introduced.
- `LASER` command improved for arbitrary intensity profile.
- Dynamic allocation (command `ALLOCATE`) of some arrays introduced. This change invokes Fortran90.

A.10 History until the version CAIN2.1b

<u>New entries on physics</u>

- 2D differential luminosity $d\mathcal{L}/dE_1dE_2$ added.

- Lorentz transformation of lasers has been added.
- Field-strength dependence of the anomalous magnetic moment of electron is taken into account in solving the Thomas-BMT equation.
- Polarization dependence of the beamstrahlung and the coherent pair creation has been included.
- The kinematics in nonlinear QED subroutines was improved so as to accept non-relativistic electrons/positrons.
- The final polarization of electron in the nonlinear Compton scattering was added.
- Polarization change in linear and nonlinear QED, beamstrahlung and coherent pair creation processes when event generation is rejected is now taken into account.
- Incoherent e^+e^- pair creation by Breit-Wheeler, Bethe-Heitler, and Landau-Lifshitz processes has been added.
- Luminosity with full polarization information (including linear polarization) can be computed.
- Differential luminosities with unequal-space energy bins are introduced.

<u>New entries on user interface</u>

- Following pre-defined variables have been added:
  `Kind`, `Gen`
- Following pre-defined functions have been added:
```
 Min, Max,
 AvrT, AvrX, AvrY, AvrS, SigT, SigX, SigY, SigS,
 AvrEn, AvrPx, AvrPy, AvrPs, SigEn, SigPx, SigPy, SigPs,
 TestT, TestX, TestY, TestS, TestEn, TestPx, TestPy, TestPs,
 LumP, LumWP, LumWbin, LumWbinEdge,
LumEE, LumEEbin, LumEEbinEdge, LumEEH, LumEEP,
 BesJ, DBesJ, BesK13, BesK23, BesKi13, BesKi53,
 FuncBS, FuncCP, IntFCP
```
- Do-type sequence in `PRINT/WRITE` command became possible.
- Maximum number of characters of user parameters is increased to 16. Also, the underscore '`_`' is allowed in parameter names.
- The flags for beamstrahlung and coherent pair creation, which had been defined in the `BBFIELD` command, were moved to a new command `CFQED` (constant-field QED). This is more logical becuase **CAIN** computes these phenomenon due to the external fields, too.[1] Acoording to this change, `CFQED` operand was added to `CLEAR` command. Except for this change, input data files prepared for **CAIN**2.3 can be used for **CAIN**2.35.

[1]Following this logic more faithfully, **CAIN** should have adopted the word `SYNCHROTRONRADIATION` instead of `BEAMSTRAHLUNG`.

- New commands `STORE` and `RESTORE` were added. You can store the variables and the luminosity data for later jobs.
- Command `PLOT FUNCTION` was added.

Bugs (already fixed in the present version)

- There was a bug in `CLEAR BEAM` command when applied during a `PUSH ENDPUSH` loop. Fixed.
- A bug was found in the file `source/physics/bb/bbmain/bbkick.f` in solving the equation of motion under the beam-beam force. It is a kind of double counting of the beam-beam effect. Fixed.
- Several bugs were found in `DRIFT, EXTERNAL` command. Fixed.
- There was a miss-spelled variable in subroutine `EVUFN` (in the directory `source/control/deciph`). (This has been overlooked because of missing `IMPLICIT NONE`.) Not very harmful. Fixed.
- Total helicity luminosity was not calculated, although differential helicity luminosity was. (A bug in `physics/lum/dlumcal.f`) Fixed.
- `PRINT/WRITE` command did not correctly understand the `KIND` operand. Fixed.
- The polarization sign of the final positron in the subroutine for linear Breit-Wheeler process (`source/physics/laser/nllsr/lnbwgn.f`) was wrong. Corrected.
- Linear polarization of final photons in the linear Compton scattering was wrong. (ξ_1, ξ_3) used to come out as $(-\xi_3, \xi_1)$. Fixed.

Bibliography

[1] K .Yokoya, *A Computer Simulation Code for the Beam-Beam Interaction in Linear Colliders*, KEK report 85-9, Oct. 1985. 6

[2] 6

[3] V. B. Berestetskii, E. M. Lifshitz and L. P. Pitaevskii, *Quantum Electrodynamics*, volume 4 of *Course of theoretical Physics*, second edition translated. Pergamon Press. 82, 100, 100, 100, 101, 105

[4] H. A. Tolhoek, Rev. Mod. Phys. **28** (1956) 277. 101

[5] Y. S. Tsai, SLAC-PUB-5924 Nov. 1992. 106

[6] T. Tauchi, K. Yokoya and P. Chen, Part. Acc. **41** (1993) 29.

[7] Yabo Liu and Ping He. 44 99

Index