

# User's Manual of **CAIN**

Version 2.44b      Feb.10.2017

TeXed on May 16, 2017

# Contents

<b>1</b>	<b>Introduction</b>	<b>6</b>
1.1	General Structure of Input Data . . . . .	7
<b>2</b>	<b>Basic Grammer of the Input Data</b>	<b>10</b>
2.1	System of Units . . . . .	10
2.2	Characters . . . . .	10
2.3	File Lines and Command Blocks . . . . .	10
2.4	Commands . . . . .	11
2.5	Expressions . . . . .	12
2.5.1	Operators . . . . .	13
2.5.2	Pre-defined parameters . . . . .	14
2.5.3	User-defined parameters . . . . .	15
2.5.4	Predefined functions . . . . .	15
2.5.5	Arrays . . . . .	16
2.5.6	Character expression . . . . .	16
2.6	<b>CAIN</b> functions . . . . .	18
2.6.1	Beam statistics functions . . . . .	18
2.6.2	Test particle functions . . . . .	19
2.6.3	Beamline functions . . . . .	20
2.6.4	Luminosity-related function . . . . .	21
2.6.5	Laser-related function . . . . .	21
2.6.6	Special functions . . . . .	21
2.7	Meta-expression . . . . .	22
2.8	External Files . . . . .	22
<b>3</b>	<b>Commands</b>	<b>24</b>
3.1	<b>ALLOCATE</b> . . . . .	25
3.2	<b>FLAG</b> . . . . .	25
3.3	<b>SET</b> . . . . .	26
3.4	<b>ARRAY</b> . . . . .	26
3.5	<b>BEAM</b> . . . . .	27
3.5.1	Definition by Twiss parameters . . . . .	27
3.5.2	Read particle data from a file . . . . .	30
3.5.3	Single particle . . . . .	33
3.5.4	Test particles . . . . .	33

3.5.5	Caution	34
3.6	LASER	34
3.6.1	General laser parameters	35
3.6.2	Time profile parameters	36
3.6.3	Spatial profile parameters	36
3.6.4	File format	37
3.6.5	Laser-related CAIN functions	40
3.7	LASERQED	40
3.8	CFQED	42
3.9	BBFIELD	43
3.10	EXTERNALFIELD	44
3.11	LUMINOSITY	44
3.12	PPINT	47
3.13	PUSH, ENDPUSH	48
3.14	DRIFT	49
3.15	LORENTZ	50
3.16	MAGNET	51
3.17	BEAMLINE	52
3.18	BLOPTICS	53
3.19	MATCHING	53
3.20	TRANSPORT, ENDTRANSPORT	55
3.21	DO, CYCLE, EXIT, ENDDO	57
3.22	IF, ELSEIF, ELSE, ENDF	59
3.23	WRITE, PRINT	59
3.23.1	Write the macro-particle data	60
3.23.2	Write the beam statistics data	61
3.23.3	Write the calculated luminosity	61
3.23.4	Write a list of defined magnets	61
3.23.5	Write the beamline optics	62
3.23.6	Write the beamline geometry	62
3.23.7	Write the values of parameters and expressions	62
3.23.8	Write a list of all allocated arrays	64
3.23.9	Write the cpu time	64
3.24	PLOT	64
3.24.1	Histogram of particle data	64
3.24.2	Scatter plot of particles or laser photons	66
3.24.3	Plot the test particle data	67
3.24.4	Plot the differential luminosity	68
3.24.5	Plot charge distribution and beam-beam field	68
3.24.6	Plot beamline optics	69
3.24.7	Plot beamline geometry	69
3.24.8	Plot a function	70
3.25	CLEAR	70
3.26	FILE	72
3.27	HEADER	73

3.28	STORE and RESTORE	73
3.29	DEBUGFLAG	74
3.30	STOP	74
3.31	END	74
3.32	Particle selection operand	74
<b>4</b>	<b>Installation</b>	<b>77</b>
4.1	UNIX Version	77
4.1.1	Directory Structure	77
4.1.2	Compilation	77
4.1.3	Storage Requirements	79
4.1.4	Run	79
4.2	Windows Version	80
4.2.1	Installation	80
4.2.2	Directory Structure	80
4.2.3	Run	80
4.2.4	Difference of usage from UNIX version	80
4.2.5	TopDrawer	81
<b>5</b>	<b>Physics and Numerical Methods</b>	<b>82</b>
5.1	Coordinate	82
5.2	Particle Variables	82
5.2.1	Arrays for Particles	82
5.2.2	Description of Polarization	83
5.3	Beam Parameters	87
5.4	Solving Equation of Motion	88
5.4.1	Equation of motion under DRIFT EXTERNAL command	89
5.4.2	Equation of motion under PUSH command	90
5.5	Beamline	91
5.5.1	Beamline Coordinate	91
5.5.2	Beamline coordinate	91
5.5.3	Dipole Magnets	92
5.5.4	Quadrupole Magnets	94
5.6	Luminosity	94
5.6.1	Luminosity Integration Algorithm	94
5.6.2	Polarization	95
5.7	Beam Field	97
5.8	Laser	100
5.8.1	Laser Geometry	100
5.8.2	Linear Compton Scattering	104
5.8.3	Compton Process in a Strong Laser Field	107
5.8.4	Breit-Wheeler Process in a Strong Laser Field	111
5.9	Beamstrahlung	114
5.9.1	Basic formulas	114
5.9.2	Algorithm of event generation	115

5.9.3	Polarization . . . . .	116
5.9.4	Enhancement factor of the event rate . . . . .	118
5.10	Coherent Pair Creation . . . . .	118
5.10.1	Basic formulas . . . . .	118
5.10.2	Algorithm of event generation . . . . .	119
5.11	Incoherent Processes . . . . .	121
5.11.1	Breit-Wheeler Process . . . . .	122
5.11.2	Virtual (almost real) photon approximation . . . . .	124
5.11.3	Numerical methods . . . . .	125
<b>A</b>	<b>History of Revision</b> . . . . .	<b>128</b>
A.1	CAIN2.44b . . . . .	128
A.2	CAIN2.43 . . . . .	128
A.3	CAIN2.42 . . . . .	128
A.4	CAIN2.40 . . . . .	128
A.5	CAIN2.35 . . . . .	129
A.6	CAIN2.33 . . . . .	129
A.7	CAIN2.32 . . . . .	130
A.8	CAIN2.31 . . . . .	130
A.9	CAIN2.3 . . . . .	131
A.10	CAIN2.23 . . . . .	131
A.11	CAIN2.21 . . . . .	132
A.12	CAIN2.2a . . . . .	132
A.13	CAIN2.2 . . . . .	132
A.14	History until the version CAIN2.1b . . . . .	132

# 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  $\gamma\text{-}\gamma$  colliders.

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

The first version **CAIN1.1**[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, **CAIN2.0** was written from scratch. It now allows any mixture of  $e^-$ ,  $e^+$ ,  $\gamma$  and lasers, and multiple-stage interactions. The input data format has been refreshed completely.

The physical objects which appear in the present version **CAIN2.44b** 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 **CAIN2.23** you can track a beam through a beamline consisting of magnets.

The physical processes that can be handled by the present version **CAIN2.44b** are

- Classical interaction (orbit deformation) due to the Coulomb field.
- Luminosity between beams ( $e^- e^+ \gamma$ ).
- 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  $\gamma\text{-}\gamma$  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 `cain244b/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  $1 \times 10^{10}$ , energy 500GeV, bunch length  $100\mu\text{m}$ , 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 bizarre. 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 .....;
```

$\gamma\text{-}\gamma$  collider is more complex. Three steps,  $e\text{-}\gamma$  conversion of right-going electron, that of left-going electron, and  $\gamma\text{-}\gamma$  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 `cain244b/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- $\gamma$  interactions has to be specified by LASERQED command:

```

LASERQED  COMPTON, NPH=5, XIMAX=1.1*xi, LAMBDA MAX=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  $\gamma$ - $\gamma$  collision.

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

# Chapter 2

## Basic Grammar 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  $\text{cm}^{-2}\text{sec}^{-1}$ . 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,

<sup>1</sup> 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, ... operand ;
```

After the `command_name` before the first operand, there must be at least one blank character (unless there is no operand). Operands are separated by a comma “,” and the number of blanks before and after “,” is arbitrary. (In some commands, “,” can be replaced by one or more blanks). 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, ... 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

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

---

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

<b>CFQED</b>	Parameters for the interaction between particles and constant electromagnetic field (beamstrahlung and coherent pair creation). Sec.3.8.
<b>BBFIELD</b>	Method of calculation of the beam field. Sec.3.9.
<b>PPINT</b>	Incoherent particle-particle interaction. Sec.3.12.
<b>LUMINOSITY</b>	Define what sort of luminosities to be calculated. Sec.3.11.
<b>LORENTZ</b>	Lorentz transformation. Sec.3.15.
<b>MAGNET</b>	Define a magnet for beamline transportation. Sec.3.16.
<b>BEAMLINE</b>	Define configuration of a beamline. Sec.3.17.
<b>MATCHING</b>	Optics matching of a beamline. Sec.3.19.
<b>BLOPTICS</b>	Calculate Twiss parameters of a beamline. Sec.3.18
<b>TRANSPORT,ENDTRANSPORT</b>	Loop for beam transportation along a beamline. Sec.3.20
<b>DRIFT</b>	Move particles in vacuum or in external field. Sec.3.14.
<b>PUSH,ENDPUSH</b>	Loop of time steps. Sec.3.13.
<b>DO,CYCLE,EXIT,ENDDO</b>	Do loop. Sec.3.21.
<b>IF,ELSEIFELSE,ENDIF</b>	If block. Sec.3.22.
<b>WRITE,PRINT</b>	Print on screen or on a file. Sec.3.23.
<b>PLOT</b>	Plot using TopDrawer. Sec.3.24.
<b>CLEAR</b>	Clear data or disable commands. Sec.3.25.
<b>FILE</b>	Open/close files. Sec.3.26.
<b>HEADER</b>	Define the header for graphic outputs. Sec.3.27.
<b>STORE,RESTORE</b>	Save/recall variables and luminosity values. Sec.3.28.
<b>STOP</b>	Stop run. Sec.3.30.
<b>END</b>	End of the input file. Sec.3.31.

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 “ $\wedge$ ” 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  $\text{SET } x=1/5, y=(5*x>=1)$ ; 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,  $\text{SET } x=3, y=5, z=(x+2>=y)$ ; 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:

$\wedge, (*, /), (+, -), (<, >, <=, >=), (==, <>), \&\&, ||$ .

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  $\text{keyword}=\text{operator}$ , 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:

<b>Pi</b>	$\pi$
<b>E</b>	$e = 2.718\dots$
<b>Euler</b>	Euler's constant $\gamma_E = 0.577\dots$
<b>Deg</b>	$\pi/180 = 0.0174\dots$ You can write, e.g., $10*\text{Deg}$ where the radian unit is required.
<b>Cvel</b>	Velocity of light (m/sec).
<b>Hbar</b>	Planck's constant (Joule·sec).
<b>Hbarc</b>	Planck's constant times the velocity of light (eV·m).
<b>Emass</b>	Electron mass (eV/c <sup>2</sup> ).
<b>Echarge</b>	Elementary charge (Coulomb).
<b>Reclass</b>	Classical electron radius (m).
<b>LambdaC</b>	Compton wavelength (m).
<b>FinStrC</b>	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 definite 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 definite meanings only in loop statements over particles (for example, in defining the axes for plots).

<b>Time</b>	Running variables for global time coordinate (m). Makes sense only inside PUSH-ENDPUSH loop.
<b>T,X,Y,S</b>	Running variables for particle coordinate (m).
<b>En,Px,Py,Ps</b>	Running variables for energy-momentum (eV, eV/c). The energy is <b>En</b> but not <b>E</b> .
<b>Sx,Sy,Ss</b>	Electron/positron spin. Helicity may be written approximately as $Ss*Sgn(Ps)$ .
<b>Xi1,xi2,xi3</b>	Photon Stokes parameters $\xi_1, \xi_2, \xi_3$ .
<b>Kind</b>	Particle species. 1,2,3 for photon, electron, positron.
<b>Gen</b>	Particle generation.
<b>Wgt</b>	Particle weight. (One macro-particle represents <b>Wgt</b> real particles.
<b>Incp</b>	1 if the particle is created by an incoherent process. Otherwise 0.
<b>\$PName</b>	Particle name. 4 bytes. Normally blank. 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.

<b>Ln,Lij</b>	(n=0,1,2,3,4, i,j=0,1,2,3) Luminosity values used in PLOT LUMINOSITY command.
<b>W</b>	Center-of-mass energy used in PLOT LUMINOSITY command.
<b>Sb1</b>	s-coordinate in a beamline. Makes sense only inside TRANSPORT-ENDTRANSPORT loop.
<b>\$PrevMag</b>	Character. Name of the previous magnet. Valid only during a TRANSPORT-ENDTRANSPORT loop.
<b>\$NextMag</b>	Character. Name of the next magnet. Valid only during a TRANSPORT-ENDTRANSPORT loop.
<b>\$InFilePath</b>	Full path of the input file directory. (includes the last character '/' (UNIX) or '\' (Windows)).
<b>\$InFileName</b>	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

<b>MsgFile</b>	File reference number for echo, error messages, and default destination of PRINT command. (default=6) <sup>2</sup>
<b>OutFile</b>	File reference number for voluminous outputs. The default destination of WRITE command. (default=12)
<b>OutFile2</b>	Other print output. Not used. (default=12)
<b>TDFile</b>	TopDrawer file number. (default=8)
<b>MsgLevel</b>	Message level. (default=0, i.e., error messages only)
<b>Rand</b>	Random number seed. Positive odd integer other than 1, default=3. You can reset random number at any time.
<b>Debug</b>	Debug parameter for the programmer. If you set <code>Debug</code> ≥ 2, call and return from major subroutines are announced. (default=0)
<b>Smesh</b>	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,**

---

<sup>2</sup>The input file number is set to 5. If you want to change it, see the variable RDFL in the file 'cain244b/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**:

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

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.<sup>3</sup> 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 recommended (there can be bugs). Even if you have both ' and " in a string, you can write like "'"+'".

Character expressions have been introduced since **CAIN2.3**. All expressions can be classified into two types, floating and character, according to the results.

---

<sup>3</sup>When a file line is read from a file, blank spaces to the end of the line cause a problem. There is no way in standard FORTRAN to distinguish between blank 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.<sup>4</sup> 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);

---

<sup>4</sup> 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($a, n1, n2)` is the substring of `$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, $b)` searches for the first occurrence of the string `$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:

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

### 2.6.1 Beam statistics functions

The number of particles, that of macro particles, the average coordinates/energy-momentum/polarization 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`,

`AvrSx,AvrSy,AvrSs,AvrXi1,AvrXi2,AvrXi3,  
SigSx,SigSy,SigSs,SigXi1,SigXi2,SigXi3,  
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,<sup>5</sup> 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.3.32 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`.

The variables (`Sx,Sy,Ss`) are used for the spin of electrons/positrons and those (`Xi1,Xi2,Xi3`) for polarization of photons. However, these two sets are not actually distinguished. For example, `AvrSs(1,2)` means average electron helicity but it is equivalent to `AvrXi3(1,2)`.

`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.)

<sup>5</sup>The meaning of this argument has changed since `CAIN2.3`. It used to be selecting the range of `S` coordinate but has been replaced by a more powerful one.

### 2.6.3 Beamline functions

The functions related to the beamline optics such as  $\beta$  and  $\eta$  functions and the transfer matrix are retrieved by

**Beta, Alpha, Eta, Etaprime, Nu, TMat**

(Nu is the phase advance  $/(2\pi)$  from the beamline entrance. **TMat** is the  $6 \times 6$  linear transfer matrix from the beamline entrance.) Prior to use, you must compute the optics by **BLOPTICS** command. The calling sequence of these functions are the same except for **TMat**:

**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 occurrence 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 occurrence 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.).

The calling sequence of **TMat** is

**TMat**(*i,j*, mag , 'bl\_name')

where

**i,j**  $1 \leq i, j \leq 6$ . Identify the matrix element.

**mag, bl\_name** Same as Twiss parameters.

caution: The 5th column/row of the transfer matrix corresponds to the time variable (multiplied by the velocity of light). It is negative at the bunch head, because the head comes to a given machine point earlier. The transfer matrix is symplectic only when  $(5, i)$  and  $(i, 5)$  ( $i \neq 5$ ) elements are multiplied by -1.

## 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<sup>2</sup> 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,  
 $j=0$ :  $\tau - \zeta$ ,  $j=1$ :  $\xi$ ,  $j=2$ :  $\eta$ ,  $j=3$ :  $\zeta$   
See Sec.5.8.1 for the laser coordinate ( $\tau, \xi, \eta, \zeta$ ).

## 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  $K_\nu$  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**( $\nu, x, k$ ) Modified Bessel function  $K_\nu(x)$ . ( $x > 0$ )

**DBesK**( $\nu, x, k$ ) 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.(5.147) 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**( $x, \Upsilon$ ) Beamstrahlung function  $F_{00}$  defined in eq.(5.145).  $x$  ( $0 < x < 1$ ) is the photon energy in units of the initial electron energy.  $\Upsilon > 0$ .

**FuncCP**( $x, \chi$ ) Spectrum function  $F_{CP}$  of coherent pair creation defined in eq.(5.161).  $x$  ( $0 < x < 1$ ) is the positron energy in units of the initial photon energy.  $\chi > 0$ .

**IntFCP( $\chi, k$ )** Integral of **FuncCP**( $x, \chi$ ) 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  $\sigma_x$  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 **CAIN2.3** (**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 referred to only by unit numbers defined by the variables **MsgFile**, **OutFile**, **OutFile2**, and **TDFile**. These unit numbers should be as-

associated 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 assigned to a **CAIN** variable but is fixed to 5. If you want to change it, you have to change the FORTRAN variable **RDFL** in 'cain244b/src/initlz.f' and compile this file.

### Other output files

Some commands such as **WRITE** and **BEAM** accept I/O files specified 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`      `op1, op2, ... , opn` ;

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

‘`opj`’ 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 keyword, 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 blank 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 unambiguous. 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 **CAIN2.2**. 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 preceded 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=  $m_{\mathcal{L}}$ , ]   [MWNLLSR=  $m_{nllsr}$ , ] ;
```

$m_p$	Maximum number of macro particles. Default is $10^5$ .
$m_{vph}$	Maximum number of virtual photons in a time step in a longitudinal slice. Default is $m_p/10$ .
$m_{mag}$	Maximum number of magnets to be defined in <b>MAGNET</b> command. Default is 200.
$m_{bl}$	Maximum number of beamlines to be defined in <b>BEAMLINE</b> 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.
$m_{nllsr}$	Work area for LSRQED (nonlinear laser QED). Default is 2000000.

### 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

<b>ECHO</b>	input data echo (default=ON)
<b>SPIN</b>	include spin calculation (default=ON) (Sorry, spin calculation cannot be avoided consistently in the present version.)
<b>EXIT</b>	CAIN normally stops with a dialog window asking stay on the console or not. If EXIT flag is on, CAIN exits without dialog window. (default=OFF) (Since CAIN2.44)

### 3.3 SET

Defines parameters.

Syntax:

**SET**      $[p = a]$     $[, p = a]$     $\dots$    ;

- 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**      $a([l_1:]n_1, [l_2:]n_2, \dots, [l_m:]n_m) [=v], \dots$  ;

- 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, a1 [, a2] ... ;
```

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  $s=\text{constant}$  (race-goal picture), rather than on the  $t=\text{constant}$  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= $N_p$ , EO= $E_0$ ,
  [TXYS=( $t, x, y, s$ ),] BETA=( $\beta_x, \beta_y$ ), [ALPHA=( $\alpha_x, \alpha_y$ ),] [EMIT=( $\epsilon_x, \epsilon_y$ ),]
  [SIGT= $\sigma_t$ ,] [SIGE= $\sigma_\epsilon$ ,] [GCUT=( $n_x, n_y$ ),] [GCUTT= $n_t$ ,] [GCUTE= $n_\epsilon$ ,]
  [GAUSSWEIGHT= $i_g$ ,] [ELLIPTIC,|JUNIFORM,] [TUNIFORM,] [EUNIFORM,]
  [SLOPE=( $\theta_x, \theta_y$ ),] [CRAB=( $\psi_x, \psi_y$ ),] [ETA=( $\eta_x, \eta_y$ ),]
  [ETAPRIME=( $\eta'_x, \eta'_y$ ),] [ESLOPE= $d\epsilon/dt$ ,] [XYROLL= $\phi_{xy}$ ,]
  [DALPHADE=( $d\alpha_x/d\epsilon, d\alpha_y/d\epsilon$ ),] [DALPHADT=( $d\alpha_x/dt, d\alpha_y/dt$ ),]
  [SPIN=( $\zeta_x, \zeta_y, \zeta_s$ ),] ;
```

**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)

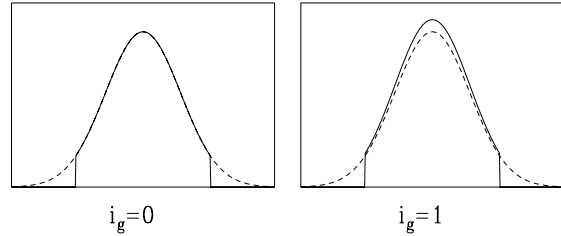
**$t, x, y, s$**  Location of the reference point and the time when the beam center comes there. In units of meter. This is the point where the Twiss parameters are to be defined. Default=(0,0,0,0).

**$\beta_x, \beta_y$**  Beta functions (m).

**$\alpha_x, \alpha_y$**  Alpha functions. Default=(0,0). The sign of  $\alpha$  is positive when the beam is going to be focused, regardless the beam is right-going or left-going.

$\epsilon_x, \epsilon_y$	R.m.s geometric emittance (rad·m). Deafault=(0,0).
$\sigma_t$	R.m.s. bunch length (m). Default=0.
$\sigma_\epsilon$	Relative r.m.s. energy spread. Default=0.
$n_x, n_y, n_t, n_\epsilon$	Gaussian tail cutoff in units of corresponding sigmas. The default values are 3. for $n_x$ and $n_y$ , $n_t$ and $n_\epsilon$ . (For transverse variables the cut off is done in the action variable, which means $J_i/\epsilon_i \leq n_i^2/2$ ( $i=x,y$ )).
$i_g$	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. <b>CAIN</b> 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



<b>ELLIPTIC</b>	Uniform transverse distribution. (Default is Gaussian.) $(x, y)$ distribution is a uniform ellipsis 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 $\sigma_{x,y}$ . <b>ALPHA</b> and <b>GCUT</b> are not used.
<b>JUNIFORM</b>	Uniform distribution for the transverse action variables $(J_x, J_y)$ . Their maximums are given by $(2\epsilon_x, 2\epsilon_y)$ .
<b>TUNIFORM</b>	Uniform $t$ -distribution. (Default is Gaussian.) The full length is $2\sqrt{3}\sigma_t$ . <b>GCUTT</b> is not used.
<b>EUNIFORM</b>	Uniform $E$ -distribution. (Default is Gaussian.) The full relative energy spread is $2\sqrt{3}\sigma_\epsilon$ . <b>GCUTE</b> is not used.
$\theta_x, \theta_y$	Angle offset (radian). The right and left-going beams have the same sign of slope when there is a crossing angle. Default=(0,0).

- $\psi_x, \psi_y$  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  $\phi_{cross}$  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  $t=\text{constant}$  (snap shot). **NONEWPAGE** operand suppresses page break so that the  $(s,x)$  profiles at different times appear on the same page.
- $\eta_x, \eta_y$  Eta function (m).
- $\eta'_x, \eta'_y$  Derivative of eta function.
- $d\varepsilon/dt$  Coherent energy slope from bunch head to tail (1/m).
- $\phi_{xy}$  Roll angle of the beam in the  $x$ - $y$  plane. (radian)
- $d\alpha_x/d\varepsilon, d\alpha_y/d\varepsilon$  Energy dependence of  $\alpha_{xy}$  (dimension less). This defines the energy dependence of the focal point. The focal point in the  $x(y)$  plane for particles with relative energy deviation  $\varepsilon$  is given by  $\varepsilon\beta_{x(y)} \times d\alpha_{x(y)}/d\varepsilon$  (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 using energy slope.
- $d\alpha_x/dt, d\alpha_y/dt$  Bunch position dependence of  $\alpha_{xy}$  (1/m). This defines the bunch position dependence of the focal point. The focal point in the  $x(y)$  plane for particles with the position in the bunch  $t$  (note:  $t > 0$  at bunch tail) is given by  $t\beta_{x(y)} \times d\alpha_{x(y)}/dt$  (positive if focused beyond the normal focal point). This operand can be used for the travelling focus scheme. See the note below.
- $\zeta_x, \zeta_y, \zeta_s$  Polarization vector. Default=(0,0,0). Note the sign of  $\zeta_s$  for left-going particles. In the case of photon beams, these are the Stokes parameter  $(\xi_1, \xi_2, \xi_3)$ . The basis vector of the Stokes parameter is  $(\mathbf{e}^{(1)}, \mathbf{e}^{(2)}, \mathbf{e}^{(3)})$  where  $\mathbf{e}^{(3)}$  is the unit vector along the particle momentum,  $\mathbf{e}^{(1)}$  is the unit vector along  $\mathbf{e}_x - \mathbf{e}^{(3)}(\mathbf{e}^{(3)} \cdot \mathbf{e}_x)$ , and  $\mathbf{e}^{(2)} = \mathbf{e}^{(3)} \times \mathbf{e}^{(1)}$ .

See Sec.5.3 for rigorous definitions.

#### Traveling Focus

A little complicated parameter definition is needed for travelling focus. For the scheme using the energy slope in the bunch is straight forward by using DALPHADE. Here is the parameter definition for the scheme without using the energy slope (e.g., crab cavity plus sextupole).

```
SET      s0=Sqrt(3)/2*sigz, dalphdty=-0.5/betay;
BEAM  RIGHT, TXYS=(-s0,0,0,-s0), .....
        DALPHADT=(0, dalphadty),.....;
BEAM  LEFT,  TXYS=(-s0,0,0,s0),
        DALPHADT=(0, dalphadty),.....;
```

where `sigz` is the bunch length and `betay` is the vertical beta function at the collision point. This sequence will make every part of the bunch to be focused when it encounters with the ‘head’ of the on-coming bunch. The definition of the ‘head’ is somewhat arbitrary. Here it is defined as  $\sqrt{3}/2\sigma_z$  (as suggested by uniform distribution) but the coefficient should be tweaked so as to give the maximum luminosity. (In fact it should be zero for zero intensity beams.)

### 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 difference 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. Otherwise 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 blank 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  $\sqrt{\mathbf{p}^2 + m^2}$ . 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 blank 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 understand 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<sub>j</sub>** 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  

$$\text{KEYWORD}=(a, b(2)),$$
- x<sub>i</sub>, f<sub>i</sub>** Define the rule to convert the input data (**KEYWORDS**) into the **CAIN** variables. The l.h.s. **x<sub>i</sub>** 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<sub>i</sub>** is an expression including the **KEYWORDS** **kw<sub>j</sub>**'s (and possibly other variables). You have to define all the 16 variables above. (Default is all zero except **Kind=1, Gen=1**.) Note that **En** does not appear here because the energy is computed from  $\sqrt{m^2 + \mathbf{p}^2}$ .

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  $p_0=1\text{GeV}/c$ . 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** variable 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

- Blank spaces in the file data are treated as delimiters. Comma ',' and line feed are replaced by blank 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 recommended, 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=(px,py,ps),  [SPIN=(sx,sy,ss),] ;
```

- 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<sub>x</sub>,p<sub>y</sub>,p<sub>s</sub>*    3-momentum (eV/c).  $p_s$  must not be zero (i.e., either right-going or left-going). The energy is computed from  $E^2 = m^2 + \mathbf{p}^2$ .
- s<sub>x</sub>,s<sub>y</sub>,s<sub>s</sub>*    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 referred 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=(px,py,ps),  [SPIN=(sx,sy,ss),] ;
```

- 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<sub>x</sub>,p<sub>y</sub>,p<sub>s</sub>*    3-momentum (eV/c).  $p_s$  must not be zero (i.e., either right-going or left-going).
- s<sub>x</sub>,s<sub>y</sub>,s<sub>s</sub>*    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  $t = \text{Time}$ . 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  $\text{TXYS} = (\text{Time}, \dots)$ , 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 4 categories.

- (a) General parameters such as wavelength, peak power density, and those defining laser coordinate sytem  $(\tau, \xi, \eta, \zeta)$ . (See Sec.5.8.1 for laser coordinate). This category includes the parameters RIGHT/LEFT, POWERDENSITY, WAVELENGTH, TXYS, E3, E1, and STOKES. Also included in this category are parameters for plotting the intensity profile.
- (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.
- (d) Parameters for file reading data. You can specify (b) and/or (c) by a file defined by the FILE operand. The parameters FILE, SHIFTT and SHIFTZ belong to this category.

The laser profile is determined in the following order.

1. FILE operand has the highest priority. If time or space profile is missing in the file, it is expected to be given by the command parameters of the category (b) or (c).
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.

The laser power density  $P$  is a function of  $(\tau, \xi, \eta, \zeta)$ . It is written as

$$P(\tau, \xi, \eta, \zeta) = P_{00} \times F_t(\tau - \zeta) \times F_s(\xi, \eta, \zeta) \quad (3.1)$$

$P_{00}$  is given in the category (a),  $F_t$  in (b) and  $F_s$  in (c). (It is also possible to specify  $F_t \times F_s$  by a file as a 4-argument function.) The flush energy observed at  $\zeta=\text{const}$  is

$$A = \int P(\tau, \xi, \eta, \zeta) d\tau / c d\xi d\eta = P_{00} \times \int F_t(\tau - \zeta) d(\tau - \zeta) / c \times \int F_s(\xi, \eta, \zeta) d\xi d\eta \quad (3.2)$$

Care must be taken for the unit system.  $P$  must be given in Watt/m<sup>2</sup>. <sup>1</sup>

### 3.6.1 General laser parameters

Syntax:

```
LASER      [RIGHT|LEFT,]  WAVELENGTH= $\lambda_L$ ,  POWERDENSITY= $P_{00}$ ,
            [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=( $\xi_1, \xi_2, \xi_3$ ),]  [PLOTPROFILE,]  [TPROFILE= $\tau_{prof}$ ] |
            [TPROFILE=( $\tau_{prof1}, \tau_{prof2}$ ), NPROFILE= $n_{prof}$ ,]  [time-profile-params,]
            [spatial-profile-params,]  [file_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.

$\lambda_L$  Laser wavelength (m).

$P_{00}$  Peak power density (Watt/m<sup>2</sup>). If FILE operand is used,  $P_{00}$  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  $e^{(3)}$  along the direction of laser propagation.

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

$\xi_1, \xi_2, \xi_3$  Stokes parameter defined in the  $(e^{(1)}, e^{(2)}, e^{(3)})$  frame. Default=(0,0,0).

---

<sup>1</sup>In the case when you use predefined profile (Gaussian etc),  $F_t$  and  $F_s$  are dimensionless (see below) so that  $P_{00}$  must be given in Watt/m<sup>2</sup>. When you use a file data, the only requirement is that the product  $P_{00} \times F_t \times F_s$  must be in units of Watt/m<sup>2</sup>. The factor  $1/c$  in the above expression of  $A$  may look bizzare but it is to convert  $\tau$  given in meter (CAIN's general rule) into second so that  $A$  comes out in Joule.

**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  $\tau_{prof}$ .

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)

$\tau_{prof}$  The time (multiplied by velocity of light) to take snapshot for plotting laser profile. In units of meter in the laser coordinate  $(\tau, \xi, \eta, \zeta)$ .

$\tau_{prof1}, \tau_{prof2}, n_{prof}$  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

Two cases, Gaussian and trapezoidal, are available. Use files for general shape. The two cases are distinguished by **SIGT/TTOT** operands. Either one of these has to be specified.

For Gaussian temporal profile:

Syntax:

**SIGT= $\sigma_\tau$ , [GCUTT= $n_{tcut}$ ,] ;**

$\sigma_\tau$  R.m.s. pulse length (times velocity of light) in power, not in field amplitude. (meter)

$n_{tcut}$  Cut off of longitudinal tail in units of sigmas. Default=3.5.

In this case the time profile factor is  $F_t = e^{-(\tau-\zeta)^2/2\sigma_\tau^2}$  (not  $e^{-(\tau-\zeta)^2/2\sigma_\tau^2}/\sqrt{2\pi}\sigma_\tau$ ).

For trapezoidal profile:

Syntax:

**TTOT= $\tau_{tot}$ , [TEDGE= $\tau_{edge}$ ,] ;**

$\tau_{tot}$  Total pulse length for trapezoidal time structure (meter).

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

In this case  $F_t$  is a function whose flat top value is unity.

### 3.6.3 Spatial profile parameters

Two cases, Gaussian and donuts, are available. Use files for general shape.

For Gaussian transverse profile:

Syntax:

**RAYLEIGH=( $\beta_1, \beta_2$ ), [GCUT= $n_{cut}$ ,] [TDL=( $d_1, d_2$ ),] ;**

$\beta_1, \beta_2$	Rayleigh length in $e^{(1)}, e^{(2)}$ direction for Gaussian beam. (meter)
$n_{cut}$	Cut off of Gaussian 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.

The spatial profile factor  $F_s$  is given explicitly in eq.(5.81).

For donuts-shape beam by an axicon:

Syntax:

**DONUT, AAXICON= $a_{ax}$ , BAXICON= $b_{ax}$ , FOCALLENGTH= $f$ , SIGMA0= $\sigma_0$ ,  
[RMAX= $r_{max}$ ,] [ZMAX= $\zeta_{max}$ ];**

**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  $\sigma_0$ . The power (Watt) integrated over the transverse plane is

$$P = 2\pi\sigma_0^2 P_{00} \left(1 - e^{-(a_{ax}^2 - b_{ax}^2)/2\sigma_0^2}\right)$$

$a_{ax}, b_{ax}$	Outer and inner radius of the axicon.
$f$	Focal length of the lens just after the axicon.
$\sigma_0$	Rms radius of the laser beam at the entrance of the axicon.
$r_{max}, \zeta_{max}$	Specify the range of laser field. $\sqrt{\zeta^2 + \eta^2} < r_{max}$ 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

Syntax:

**FILE= $f_n$  | 'file\_name', [SHIFTT= $\Delta\tau$ ,] [SHIFTZ= $\Delta\zeta$ ,] [ZMAX= $\zeta_{max}$ ]**

$f_n, \text{file\_name}$  Unit number or name of an existing file which describe the laser field profile. (For the difference between the two forms, See Sec.2.8.) See below (Sec.3.6.4) for the file format.

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

$\zeta_{max}$  Range of field  $-\zeta_{max} < \zeta < \zeta_{max}$  to extend the file data. Needed only for ORDER=XY (see below).

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 dependence of the power density on  $\tau - \zeta$  (L),  $\xi$  (X),  $\eta$  (Y),  $\zeta$  (Z),  $\sqrt{\xi^2 + \eta^2}$  (R). The possible combinations are

L	$F_t(\tau - \zeta)$
XYZ	$F_s(\xi, \eta, \zeta)$
RZ	$F_s(r, \zeta)$
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).

- For the case L, the spatial profile  $F_s$  is expected to be given in other place (operands of LASER command or another ORDER in the same file.
- For the case XYZ or RZ, the temporal profile  $F_t$  is expected to be given in other place (operands of LASER command or another ORDER in the same file.
- 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. Thus, the interval is  $(a_2 - a_1)/(n_a - 1)$ . The present version accepts equally spaced data only. ( $a_2$  can be  $< a_1$ .) For example, if ORDER=RZ, the lines
 
$$R = r_1 \ r_2 \ n_r$$

$$Z = \zeta_1 \ \zeta_2 \ n_\zeta$$
 are expected. (Input order of R= and Z= is irrelevant.)

- Optional lines to specify the units in the form

aFAC=c

where ‘a’ is one of the characters which appear in ORDER or ‘P’ for power density (see below) or F for phase (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<sup>2</sup>).

- Finally, power density data in the form

P = P<sub>111</sub> P<sub>211</sub> P<sub>311</sub> . . .

The data are separated by one or more blank 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<sup>2</sup> (after multiplied by PFAC in the file and/or POWERDENSITY in the command). Linear interpolation is applied.

- For the case `ORDER≠L`, you may optionally define the local propagation direction  $\mathbf{n}$  in the form for `ORDER=XYZ` or `ORDER=LXYZ`

$$\mathbf{N} = n_{\xi,111} \ n_{\eta,111} \ n_{\zeta,111} \ n_{\xi,211} \ n_{\eta,211} \ n_{\zeta,211} \ \dots$$

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

$$\mathbf{N} = n_{r,111} \ n_{\zeta,111} \ n_{r,211} \ n_{\zeta,211} \ \dots$$

If  $\mathbf{N}$  does not appear,  $\mathbf{n} = (0, 0, 1)$  is used, which means parallel propagation along E3 vector.

- Then, `ORDER` can appear again to define other dependences. For example, the first `ORDER` defines  $F_t(\tau - \zeta)$  and the second defines  $F_s(r, \zeta)$ . Of course, there can be only one `ORDER` if it defines  $F(\tau - \zeta, \xi, \eta, \zeta)$  or defines  $F(\tau - \zeta, r, \zeta)$ .

### Special case of 2D file input

Theoretically, it is possible to extend a 2D data given on  $(\xi, \eta)$  plane at  $\zeta = \text{constant}$  to 3D region  $(\xi, \eta, \zeta)$  using the Maxwell equation. (See Sec.5.8.1 for the method.) **CAIN** does this provided

- Complex phase data in addition to the intensity is given.
- Propagation in vacuum.
- Perfectly coherent beam
- The focal points in the  $\xi$ - and  $\eta$ -planes are close to each other (within the Rayleigh length) and close to the point  $\zeta$  where the 2D data is given. (This condition is only to avoid programming complication.)

The file format in this case is the following.

`ORDER=XY`

`X=  $\xi_1 \ \xi_2 \ n_\xi$`

`Y=  $\eta_1 \ \eta_2 \ n_\eta$`

`PFAC=  $c_p$`

`FFAC=  $c_\phi$`

`P=  $P_{11} \ P_{21} \ P_{31} \ \dots$`

`F=  $\phi_{11} \ \phi_{21} \ \phi_{31} \ \dots$`

- The temporal profile is expected to be given in other place (operands of `LASER` command or `ORDER=L` in the same file.
- The file data is assumed to be given on the plane  $\zeta = 0$ . If not, you can use the operand `SHIFTZ` in `LASER` command.
- To create a 3D table in  $(\xi, \eta, \zeta)$  plane, the range of  $\zeta$  must be specified. This has to be given by `ZMAX= $\zeta_{max}$`  operand of the `LASER` command. Unnecessarily large  $\zeta_{max}$  may cause memory error or be a waste of computing time.
- `F=` gives the phase data in units of radian. (You can use degree here by specifying `FFAC=0.01745329` ( $= \pi/180$ ).) The sign of the phase is defined such that the electric field is proportional to  $\sqrt{P}e^{i\phi - i\omega t}$ .

### 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  $\xi_{max}$  is needed in `LASERQED` command.

Another CAIN function is `LaserRange` to retrieve the range of coordinate 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.5.8.2, Sec.5.8.3 and Sec.5.8.4 for more detail on physics.

As of `CAIN2.44b` 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= $n_\xi$ ,] [NLAMBDA= $n_\lambda$ ,] [NQ= $n_q$ ,] XIMAX= $\xi_{max}$ ,
              LAMB DAMAX= $\lambda_{max}$ , ETAMAX= $\eta_{max}$ , [P MAX= $p_{max}$ ,] [PSTOP= $p_{stop}$ ,]
              [ENHANCEFUNCTION= $f_{enh}$ ,] [LENHANCE= $l_{enh}$ ,] [PLOTTABLE,];
```

`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 `CAIN2.44b` 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  $\geq 1$ , use nonlinear formula.

Note that  $n_{ph} = 0$  and  $n_{ph} = 1$  are different. The former is the limit of  $\xi \rightarrow 0$ , 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 ( $n_y, n_\xi, n_\lambda, n_q, \xi_{max}, \lambda_{max}, \eta_{max}$ ) are needed.

When  $n_{ph} \geq 1$ , only longitudinal polarization is considered and the lasers must be circularly polarized by 100% (i.e.,  $\xi_1 = \xi_3 = 0, \xi_2 = \pm 1$ ).

When the laser field is intense ( $\xi \gtrsim 1$ ), about  $n_{ph} \gtrsim 2\xi^3$  is needed. When  $n_{ph}$  is too large, the allocated memory would not be sufficient and `CAIN` stops with a message. In such a case use `MWNLLSR` operand of `ALLOCATE` command.



- n<sub>y</sub>* Number of abscissa for final energy. Default=20.
- n<sub>ξ</sub>* Number of abscissa for ξ parameter. Default=20.
- n<sub>λ</sub>* Number of abscissa for λ parameter. Applies to Compton case only. Default=20.
- n<sub>q</sub>* Number of abscissa for q parameter. Applies to Breit-Wheeler case only. Default=50.
- ξ<sub>max</sub>* Maximum value of ξ for the table.
- λ<sub>max</sub>* Maximum value of λ for the table. Applies to Compton case only.
- η<sub>max</sub>* Maximum value of η for the table. Applies to Breit-Wheeler case only.
- p<sub>max</sub>* Maximum probability of events in one step. When the probability of events in the time step specified by **PUSH** command for a particle, **CAIN** temporarily subdivides the time interval for the particle such that the probability is less than *p<sub>max</sub>*. Default is *p<sub>max</sub>* = 0.5.<sup>2</sup>
- p<sub>stop</sub>* Maximum probability of events per one time step specified by **PUSH** command. If the computed probability exceeds *p<sub>stop</sub>*, **CAIN** stops with a message. Default is *p<sub>stop</sub>* = 100. It is not recommended to define too large a value because the possible events by the new particles within the same time step are ignored. This operand was introduced since **cain2.42**.
- PLOTTABLE** Create some plots of the table. Mostly for debugging. Not ready for linear polarization.
- f<sub>enh</sub>* 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 ≤ Y ≤ 1). Its value must be ≥ 1 for all Y. Generally speaking, Y close to 1 generates low energy charged particles. For example,  

$$\text{ENH}=1+\text{Step}(Y-0.8)*(Y-0.8)*10$$
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<sub>enh</sub>* and, when an event is generated, the created particles are assigned a weight 1/*f<sub>enh</sub>*.<sup>3</sup>  
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<sub>enh</sub>* Integer 1 or 2 or 3. Defines how to treat macro-particles under enhancement. Needed only when you specify *f<sub>enh</sub>*. 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<sub>enh</sub>* = 1, **CAIN** generates

---

<sup>2</sup> The meaning of *p<sub>max</sub>* has been changed since **cain2.42**. Till **cain2.40** the execution used to be stopped when *p<sub>max</sub>* is exceeded.

<sup>3</sup> Note that *f<sub>enh</sub>* slightly larger than 1 is useless (even harmful) because a small fraction 1 - 1/*f<sub>enh</sub>* of the parent particle will remain as a macro-particle, causing a waste of computing time. In the example above, *f<sub>enh</sub>* = 1 exactly for Y < 0.8.

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 ( $\gamma$ ) is treated probabilistically, i.e., a  $\gamma$  of weight  $w_0$  (not  $w_0/f_{enh}$ ) is generated with the probability  $1/f_{enh}$ . Thus, the number of macro- $\gamma$  is the same as in the absence 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 ( $\gamma$ ) 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 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  $n_{macro\gamma}/n_{macroe} = n_{real\gamma}/n_{reale}$ . 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= $\Delta$ ,]    [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  $3 \times 2^n$  or  $5 \times 2^n$  are recommended. Default=32.

**$\Delta$**  Macro-particle size in units of the bin size. Macro-particles are treated as a rectangular of uniform distribution. Must be  $0 \leq \Delta \leq 1$ . 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 throughout 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=(s1,s2),]    [V=(cx,cy,cs),]    [E=(Ex,Ey,Es),]
                 [B=(Bx,By,Bs),] ;
```

- $s_j, c_j$  Define the range of the field as  $s_1 \leq c_x x + c_y y + c_s s \leq s_2$ . 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  $\gamma, e^-, e^+$ , right-going and left-going. Thus, there can be at most 9 **LUMINOSITY** commands.

Syntax:

```
LUMINOSITY      KIND=(kr,kl),    [FREP=frep,]
                [W=(Wmin,Wmax,nbin),|W=(W0,W1,...,Wnbin),|W=warray,]
                [E1=(E1min,E1max[,n1bin]),|E1=(E1,0,E1,1,...,E1,n1bin),|E1=e1array,]
                [E2=(E2min,E2max[,n2bin]),|E2=(E2,0,E2,1,...,E2,n2bin),|E2=e2array,]
                WX=(wx[,wxm]),    WY=(wy[,wym]),    [HELICITY,]    [ALLPOL,]
                [PPDATA=nf,] ;
```

- $k_r, k_l$  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.

- $W_0 \dots W_{n_{bin}}$  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  $\geq 3$  in order to distinguish from the equal-space case.  $n_{bin}$  must be  $\leq 200$ .
- $w_{array}$  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  $(W_0 \dots W_{n_{bin}})$ .
- $E_{1min}, E_{1max}, n_{1bin}, E_{2min}, E_{2max}, n_{2bin}$  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  $n_i, bin$  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.)
- $n_f$  If  $n_f > 0$ , properties of particle pairs contributing to the corresponding luminosity are written on the file of unit number  $n_f$ . The file must be opened before the PUSH command starts by using FILE OPEN command.  $n_f$  can be the same for all the luminosities. For each particle-particle collision, the following 3 lines are written on the file.  
line 1:  $k_1, k_2, \Delta L$  with format ('\*', 2I2, 1PE20.12)  
where  $k_{1(2)}$  is the particle species of right(left)-going particles and  $\Delta L$  is the contribution of this collision to the luminosity  
line 2: properties of right-going particle  
See the "standard format" in Sec.3.5.2 for the format of the particle perproperties. line 3: properties of left-going particle  
Relevant parameters for PPDATA are  $w_x, w_y$  and  $w_{xm}, w_{ym}$ . Be careful that the output can be huge.

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  $\text{cm}^{-2}\text{sec}^{-1}$ .
- `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( $k_r, k_l$ )`. ( $\text{cm}^{-2}\text{sec}^{-1}$ )
- `LumP( $k_r, k_l, s_1, s_2$ )` Polarization luminosity. ( $0 \leq s_1 \leq 3$ ,  $0 \leq s_2 \leq 3$ ) See Sec.5.6.2 for definition. ( $\text{cm}^{-2}\text{sec}^{-1}$ )
- `LumW( $k_r, k_l, n$ )` Differential luminosity in the  $n$ -th bin. ( $\text{cm}^{-2}\text{sec}^{-1}/\text{bin}$ )
- `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. ( $0 \leq n \leq$  number of bins.  $n = 0$  is the lower edge of the first bin and  $n =$  number of bins 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. ( $\text{cm}^{-2}\text{sec}^{-1}/\text{bin}$ )
- `LumWP( $k_r, k_l, n, s_1, s_2$ )` Differential polarization luminosity. ( $0 \leq s_1 \leq 3$ ,  $0 \leq s_2 \leq 3$ ) See Sec.5.6.2 for definition. ( $\text{cm}^{-2}\text{sec}^{-1}/\text{bin}$ )
- `LumEE( $k_r, k_l, n_1, n_2$ )` 2-D differential luminosity  $d\mathcal{L}/dE_1dE_2$  for the bin  $(n_1, n_2)$ . ( $\text{cm}^{-2}\text{sec}^{-1}/\text{bin}$ )
- `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.<sup>4</sup>

The following processes are included:

Breit-Wheeler	$\gamma + \gamma \rightarrow e^- + e^+$
Bethe-Heitler	$\gamma + e^\pm \rightarrow e^\pm + e^- + e^+$
Landau-Lifshitz	$e + e \rightarrow e + e + e^- + e^+$
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

<b>PPINT</b>	<b>VIRTUALPHOTON</b> [,] <sup>†</sup> [ <b>LOCAL</b> ,] [ <b>FIELDSUPPRESSION</b> ,] [ <b>EMIN</b> = $E_{min}$ ,] ;
<b>LOCAL</b>	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.
<b>FIELDSUPPRESSION</b>	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 <b>LOCAL</b> 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**[,]<sup>†</sup> [**RIGHT**,] [**LEFT**,]  
[**ENHANCE**= $f_{enh}$ ,] ;

<sup>4</sup> There is a known bug: PPINT is not effective when LUMINOSITY command is not invoked.

**BW,BH,LL,BREMSSTRAHLUNG** Specify one of Breit-Wheeler, 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,  $(\gamma, e^\pm)$  and  $(e^\pm, \gamma)$ . **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=( $t_0, t_f, n_t$ ) ;  
... 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. ( $\geq 0$ )

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 ( $j = 0, 1, \dots, n_t$ ), 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  $t_j \leq \mathbf{Time} \leq t_{j+1}$ .
- 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= $t_1$ |DT= $\Delta t$ |S= $s_1$ ,    [RIGHT,]    [LEFT,]    [KIND= $k$ |( $k_1, k_2$ ),]
        [EXTERNALFIELD,] ;
```

$t_1$             Drift until  $\mathbf{Time}=t_1$  (meter).

$\Delta t$             Drift over time interval  $\Delta t$  (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  $k$ .

EXTERNALFIELD    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 create 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=( $\Delta t, \Delta x, \Delta y, \Delta s$ ),]  [ANGLE= $\phi$ ,]  [BETAGAMMA= $\beta\gamma$ ,]
           [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)

$\phi$  Spatial rotation angle (radian). (rotation of the coordinate axis.)

$\beta\gamma$  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 `BEAMLIN` command. Here ‘MAGNET’ means any element in beamlines such as magnets, drift 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= $\theta$ ,] [EDGE=( $\epsilon_1, \epsilon_2$ ),] [K1= $k_1$ ,]
            [ROTATE= $\phi$ ,] [APERTURE=( $a_x, a_y$ ),] [ROUND,] [RECTANGULAR,] ;
```

- name** Magnet name. Must be enclosed by apostrophes.<sup>5</sup> (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.
- $\theta$**  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).
- $\epsilon_1, \epsilon_2$**  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 momentum 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.
- $\phi$**  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).
- ROUND** Elliptic aperture. (Rectangular if not specified.) If only one of  $a_x$  and  $a_y$  is specified, assume  $a_x = a_y$ .

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.

---

<sup>5</sup>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`).

- Errors of magnets.
- 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 BEAMLIN

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:

```
BEAMLIN      'name',    LINE=(name1, name2, ..., namen)
              [APERTURE=(ax, ay), ] ;
```

[ROUND]

- |                                       |   |
|---------------------------------------|---|
| <b>name</b>                           | 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.) <b>Present version does not work correctly if re-defined. Sorry.</b> |
| <b>name<sub>j</sub></b>               | Name of already defined magnet or beamline. You can put <code>-</code> sign for a reversed beamline. You cannot use nested <code>( )</code> . If needed, you must assign a name for the part of the beamline.   |
| <b>(a<sub>x</sub>, a<sub>y</sub>)</b> | 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.  |
| <b>ROUND</b>                          | Elliptic aperture. (Rectangular if not specified.) If only one of $a_x$ and $a_y$ is specified, assume $a_x = a_y$ .  |

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=( $\beta_x, \beta_y$ ),]
           [ALPHA=( $\alpha_x, \alpha_y$ ),] [ETA=( $\eta_x, \eta_y$ ),] [ETAPRIME=( $\eta'_x, \eta'_y$ ),] ;
```

**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)

$\beta_x, \beta_y, \alpha_x, \alpha_y, \eta_x, \eta_y, \eta'_x, \eta'_y$  Twiss parameters and eta functions at the entrance. The default values are all zero so that at least  $\beta_x$  and  $\beta_y$  must be defined if not PERIODIC.

### 3.19 MATCHING

Optics matching of a beamline.

Syntax:

```
MATCHING    BEAMLINE='name', [PERIODIC,] [BETA=( $\beta_x, \beta_y$ ),]
           [ALPHA=( $\alpha_x, \alpha_y$ ),] [ETA=( $\eta_x, \eta_y$ ),] [ETAPRIME=( $\eta'_x, \eta'_y$ ),]
           VARIABLE=( $v_1, v_2, \dots$ ), [ZERO=( $f_1, f_2, \dots$ ),] [POSITIVE=( $g_1, g_2, \dots$ ),]
           [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)

$\beta_x, \beta_y, \alpha_x, \alpha_y, \eta_x, \eta_y, \eta'_x, \eta'_y$  Twiss parameters and eta functions at the entrance. The default values are all zero so that at least  $\beta_x$  and  $\beta_y$  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 repeatedly 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. You can use the beamline functions (Twiss parameters, etc.) described in Sec.2.6.3.
- $g_i$*  Expressions that must be  $\geq 0$ . 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

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

**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, BEAMLIN='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 **CAIN2.3** 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  $2\pi$ . 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  $\sum f_i^2 + \sum [\theta(-g_i)g_i]^2$  ( $\theta$ : 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 tedious,

```

SET bx=1, by=1, ax=0, ay=0;
MATCHING BEAMLIN=..., 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);

```

( $\eta$  and  $\eta'$  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= $\epsilon$ 
             [RIGHT|LEFT,]    [KIND= $k$ |( $k_1, k_2$ )],    [LOSSMONITOR,] ;
```

Syntax:

```
ENDTRANSPORT ;
```

- |  |   |
|--|---|
| <b>name</b>                            | The beamline name. (You may omit apostrophes.)  |
| <b><math>t_0, x_0, y_0, s_0</math></b> | Entrance of the beamline in CAIN coordinate. (m)  |
| <b>E1,E3</b>                           | Unit vectors along the direction of the horizontal axis and the longitudinal axis of the beamline at its entrance in CAIN coordinate.   |
| <b><math>p_0</math></b>                | Reference momentum in eV/c.   |
| <b><math>\epsilon</math></b>           | $\pm 1$ . Sign of the charge of the reference particle.   |
| <b>RIGHT,LEFT</b>                      | 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 assigned 'lost'. If you specify one of them, 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 assigned 'lost'. |
| <b>KIND</b>                            | Apply to specified particle species only. Tracking photons can make sense only when aperture cut off is needed.   |



**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 frozen 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. Therefore, 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 ( $0 \leq j \leq n$ ) 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 through 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 `Sb1` 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 ( $1 \leq j \leq$  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 occurrence. For example the second occurrence of `QF` is `'QF.2'`. If you want to separate the name and the occurrence id, you can say, e.g.,

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

### Caution

- 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.)<sup>6</sup>

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 assigning false for 0 and true otherwise. For example, the do loop DO WHILE  $n$ ; ends when  $n = 0$ .  
For compatibility with old versions, The substitution operator = is treated as == in DO WHILE (and IF) command.

The loop is repeated so long as the condition is satisfied. 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=(i1,i2[,i3]);
```

*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.)

*i1,i2,i3* 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

---

<sup>6</sup> CAIN2.31 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:

```
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 assigning 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 substitution 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|GEANT4] [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 difference between the two forms, See Sec.2.8.

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

**MATHEMATICA** MATHEMATICA list style format.

**Caution:** The list style in the output file is

$\{\dots\}$ ,  $\{\dots\}$ , ...,  $\{\dots\}$ ,

*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.

**GEANT4** Geant4 format. The following 7 numbers are written in a line for each macro-particle with a space character in between (actually 7E20.12)

$(E, v_x/c, v_y/c, v_s/c, x, y, s)$ ,

where  $E$  is the energy (GeV) with the sign of charge and  $x, y, s$  are in units of nm. Photons are not written. Note: information of  $t$  does not appear so that user should make sure all the particles have the same  $t$ .

Use standard format (see Sec.3.5.2) if none of the above three are specified.

**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.3.32 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 difference between the two forms, See Sec.2.8.

### 3.23.3 Write the calculated luminosity

Syntax:

```
PRINT LUMINOSITY, KIND=( $k_1, k_2$ ) [FILE= $f_n$ ]'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 difference between the two forms, See Sec.2.8.

### 3.23.4 Write a list of defined magnets

Syntax:

```
WRITE    MAGNETS,  [BEAMLINE='name',]  [COMBINE,]  [MOMENTUM= $p_0$ ,]  
        [FILE= $f_n$ |'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',]  $x_1$ [,  $x_2$ [,  $x_3 \dots$ ]],  
[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 { }.<sup>7</sup> It is also possible to write a do-type sequence of the form (almost like FORTRAN)

$(y_1, \dots, y_n, i=i_1, i_2, i_3)$

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.<sup>8</sup>

**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.<sup>9</sup> 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');
```

---

<sup>7</sup> 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 grammar. 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.

<sup>8</sup>Here is a problem of grammar. Only the character expression should be accepted for consistency of grammar, 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.

<sup>9</sup>I-format was not allowed before **CAIN2.3**.

will cause ‘nothing’ be printed.<sup>10</sup>

Unfortunately, the grammar of **CAIN** does not allow an un-paired apostrophe so that, for example, 1H’ will cause an error.

*f<sub>n</sub>*, *file\_name*, **APPEND** Unit number or name of an old or new file. For the difference between the two forms, See Sec.2.8.

### 3.23.8 Write a list of all allocated arrays

Syntax:

**PRINT**     **ARRAY**,    [**LONG**,]    [**FILE=***f<sub>n</sub>* | ‘*file\_name*’, ] ;

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

### 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.

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

---

<sup>10</sup> A known bug. `WRITE (i=1,2), FORMAT=('nothing');` won't work. Instead, you can write `WRITE ('nothing',i=1,2);`



### 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= $f_x$ , V= $f_y$ , [SUM,]
[HSCALE=( $x_{min}, x_{max}, n_{bin}$ ),] [VSCALE=( $f_{min}, f_{max}$ ),] [HLOG,|HLINEAR,]
[VLOG,|VLINEAR,] [COLOR=color,] [ERRBAR,] [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.

$f_x$  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,

$$H = \text{Sqrt}[(Px^2 + Py^2) / Ps^2] * 1E6$$

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  $n_{bin} = 50$ .

$f_y$  HISTOGRAM is intended to plot particle population for each bin but it is also possible to plot other function averaged over each bin by specifying the function  $f_y$ .

**SUM** Effective when  $f_y$  is specified. If SUM is specified, the vertical axis is the sum of  $f_y$  of all the particles in the bin. If SUM not is specified, average of  $f_y$  in the bin is plotted. Thus, if you want a histogram where the vertical axis is, for example, the sum of energy of all the particles in the bin in units of Joule, then you should say V=En\*Echarge, SUM. The case of unspecified V is equivalent to V=1, SUM.

$f_{min}, f_{max}$  Range of the vertical axis. Normally this is not needed for plotting a histogram, but it is better to specify them explicitly if you want to plot other function by the V operand.

- 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.3.32 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) ;**.
- 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.
- ERRBAR** Flag to add error bars to the plot. The error bar is defined as the  $\sqrt{n}$ -type statistical spread for population or, when  $f_y$  is defined, r.m.s. data spread within the bin (correction due to possible variation of the function within the bin is not applied). In the case of secondary particles the statistics of their parent particles are not taken into account. The error bars so determined are often misleading so that they are drawn only when this flag is given.
- top\_title, etc** Title string. Must be enclosed by a pair of apostrophes. Topdrawer case string can be specified by using “;” as the delimiter like  
**TITLE='EOG1; XGX;'**, for writing  $E_\gamma$ . It is recommended to put “;” also at the end, as in this example, to avoid writing unnecessary blank characters.<sup>11</sup>
- $f_n, file\_name, APPEND$**  Output file unit number or name of an old or new file. If not specified, **FILE=TDfile**. For the difference between the two forms, See Sec.2.8.

---

<sup>11</sup> 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 becomes '''' in the TopDrawer input data and is understood 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|(k1,k2),] H=fx,  
V=fy, [HSCALE=(xmin,xmax),] [VSCALE=(ymin,ymax),] [COLOR=color,]  
[TITLE='top_title',] [HTITLE='bottom_title',] [VTITLE='left_title',]  
[FILE=fn|'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=(k1,k2), [FILE=fn|'file_name',] [APPEND,]  
[VLOG,|VLINEAR,] [PERBIN|PERHVAR,] [COLOR=color,];
```

**k<sub>1</sub>,k<sub>2</sub>** 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<sup>2</sup>/sec/bin) is plotted.

**PERHVAR** Luminosity per unit increment of horizontal axis (energy) is plotted. (1/cm<sup>2</sup>/sec/eV). Default is PERBIN.

More flexible plot is possible with the complicated syntax

Syntax:

```
PLOT LUMINOSITY, KIND=(k1,k2), V=f, [NONEWPAGE,]  
[VSCALE=(ymin,ymax),] [VLOG,|VLINEAR,] [PERBIN|PERHVAR,]  
[COLOR=color,] [TITLE='top_title',] [HTITLE='bottom_title',]  
[VTITLE='left_title',] [FILE=fn|'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/cm<sup>2</sup>/s/bin. (Or 1/cm<sup>2</sup>/s/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 processed. 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<sub>j</sub>* Define the s-coordinate. Plot for the slice which contains one of *s<sub>j</sub>*'s. Upto 5 *s<sub>j</sub>*'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.
- $l_1, l_2, \dots$**  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.3.32 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<sub>1</sub>', 'string<sub>2</sub>', ... are concatenated. The 'case' string for Top-  
Drawer can be defined by using semicolon ";" as delimiter, like  
HEADER 'JLC EOCM1=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.<sup>12</sup> 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 transferred to later jobs by `STORE`, `RESTORE`, and `WRITE BEAM` commands.

### 3.29 DEBUGFLAG

Set runtime flags for debugging purpose. Users should not use this command. Added since cain2.42.

Syntax:

```
DEBUGFLAG ID= i ;
```

*i* Flag ID.  $1 \leq i \leq 8$ . Cleared at start.

### 3.30 STOP

Stop `CAIN` run.

---

<sup>12</sup> Exceptions are the two variables `MsgFile` and `MsgLevel`. They are not stored and, hence, not restored.

### 3.31 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 grammar beyond **END** is not checked in contrast to **STOP**.

### 3.32 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

<b>T,X,Y,S</b>	particle time-space coordinate (m).
<b>En,Px,Py,Ps</b>	energy-momentum (eV, eV/c).
<b>Sx,Sy,Ss</b>	Electron/positron spin.
<b>Xi1,Xi2,Xi3</b>	Photon Stokes parameters $\xi_1, \xi_2, \xi_3$ .
<b>Kind</b>	Particle species. 1,2,3 for photon, electron, positron.
<b>Gen</b>	Particle generation.
<b>Wgt</b>	Macro-particle weight. One macro-particle corresponds to <b>Wgt</b> real particles.
<b>Incp</b>	Logical variable. True (=1) if the particle is created by an incoherent process. False (=0) otherwise.
<b>\$PName</b>	Macro-particle name. 4byte character. Normally blank. Incoherent particles start with 'I', e.g., 'IBW ', 'IBH ', 'ILL ', 'IBR ' for Breit-Wheeler, Bethe-Heitler, Landau-Lifshitz and bremsstrahlung, respectively.

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.<sup>13</sup> 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 **CAIN2.3** because its syntax like **GEN>2** is not compatible with the new grammar. You can now say **SELECT= Gen>2**.

---

<sup>13</sup> The evaluation of the expression  $f_{sel}$  is done by a sort of ‘compiled load module’ rather than ‘interpreter’, but still the computing time is a problem.

There is a complication on the selecting operand `INCP`, which exclusively select incoherent particles. The versions since `CAIN2.32` 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.

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 but you do not want them included. 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,

<code>AvrEn(1,1)</code>	average energy of normal, right-going photons
<code>AvrEn(1,1,'Incp==1')</code>	average energy of incoherent, right-going photons
<code>AvrEn(1,1,'')</code>	average energy of all right-going photons.

# Chapter 4

## Installation

The development platform of **CAIN** used to be a UNIX machine but since **CAIN2.2** it moved to Windows. There has been inconvenience for UNIX users. **CAIN2.35** 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 web page <http://lcdev.kek.jp/~yokoya/CAIN/>
2. Get the source code `cain244b.tar.gz` and the manual `CainMan235.pdf`

#### 4.1.1 Directory Structure

The directory structure is shown below.

cain	src	many files	most of the fortran source files.
		-unix	source files for unix (actually, dummy routines of Windows)
		-module	source files containing <b>MODULES</b> .
		-include	all the files to be <b>INCLUDED</b> .
		-obj	directory to store objects
	-exec		load module, shell scripts
	-in		input data
	-out		outputs
	-doc		documents

#### 4.1.2 Compilation

Since **CAIN2.2** 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 `cain244b/exec` contains a `cs`h 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 `cain244b/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` ‘touches’ 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 `CAIN2.35`.

### 4.1.3 Storage Requirements

Dynamic allocation of FORTRAN90 has been used since **CAIN2.2** 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.)  
 $192 \times \text{MP} = 19.2\text{MB}$  (default: MP=100000)
- MVPH** Maximum number of virtual photons in a time step in an *s*-slice.  $80 \times \text{MVPH} = 0.8\text{MB}$  (default: MVPH=10000)
- MMAG** Maximum number of magnets of different types.
- MBEAMLIN** Maximum number of beamlines.
- MBBXY** Maximum number of bins in each of *x* and *y* for beam-beam force calculation.  $88 \times \text{MBBXY}^2 = 1.37\text{MB}$ . (default: MBBXY=128)
- MLUMMESH** Maximum number of bins in each of *x* and *y* for for luminosity calculation.  $152 \times \text{MLUMMESH}^2 = 2.4\text{MB}$ . (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.  $8 \times \text{MWLUM} = 1.6\text{MB}$  (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.  
 $100 \times \text{MTSTP} = 0.5\text{MB}$  (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 `cain244b/in` with file names having the extension `.i`. The file set sent to you may contain some example data. The directory `cain244b/exec` contains a `csh` script file `@go` for execution. It works only when the current directory is `cain244b/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 `cain244b/out/example.tdr`, OutFile on `cain244b/out/example` and OutFile2 on `cain244b/out/example.out2`.

If you want a submit job, please write an appropriate 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 <b>CainW.exe</b> of release version
	—Debug		contains the load module <b>CainW.exe</b> of debug version
	—resource		contains the files to create icons.
	—in		contains example data.
	—doc		documents ( <b>readme.txt</b> )
	—other files		files of project settings. The main project file is <b>CainW.dsw</b> .

### 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` command 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. <sup>1</sup>

- 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`

(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 <sup>2</sup> 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 **CAIN2.2**, 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>

---

<sup>1</sup>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.

<sup>2</sup>This is not true with **CAIN2.32**.

# 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<sub>1</sub>** 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  $(\xi_1, \xi_2, \xi_3)$  for photons.  $(S_x, S_y, S_s)$  is defined, as usual, in particle's rest frame. Therefore, it acquires

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

For defining the Stokes parameter, one needs a set of orthonormal basis vectors  $(\mathbf{e}^{(1)}, \mathbf{e}^{(2)}, \mathbf{e}^{(3)})$  with the third vector  $\mathbf{e}^{(3)}$  parallel to the momentum. In `CAIN`, the first vector  $\mathbf{e}^{(1)}$  is taken to be the unit vector along  $\mathbf{e}_x - \mathbf{e}^{(3)}(\mathbf{e}_x \cdot \mathbf{e}^{(3)})$  and  $\mathbf{e}^{(2)} = \mathbf{e}^{(3)} \times \mathbf{e}^{(1)}$ . 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,  $(\mathbf{e}^{(1)}, \mathbf{e}^{(2)}, \mathbf{e}^{(3)})$  is almost equal to  $(\mathbf{e}_x, \mathbf{e}_y, \mathbf{e}_z)$  for right-going photons, and  $(\mathbf{e}_x, -\mathbf{e}_y, -\mathbf{e}_z)$  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 `GENs` of the parents. In this case `GENs` 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  $\varphi$ . The  $2 \times 2$  density matrix  $\rho^{(e)}$  is defined as

$$\rho_{ij}^{(e)} = \langle \varphi_i \varphi_j^\dagger \rangle, \quad (i, j = 1, 2) \quad (5.1)$$

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

$$\rho^{(e)} = \frac{1}{2}(1 + \boldsymbol{\zeta} \cdot \boldsymbol{\sigma}), \quad \boldsymbol{\zeta} = \text{Trace}(\rho^{(e)} \boldsymbol{\sigma}) = \langle \varphi \boldsymbol{\sigma} \varphi^\dagger \rangle \quad (5.2)$$

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

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

$$\varphi = c_+\varphi_+ + c_-\varphi_-, \quad |c_+|^2 + |c_-|^2 = 1. \quad (5.3)$$

With the standard representation of the Pauli matrices

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

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

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

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  $\mathbf{e}$  ( $|\mathbf{e}| = 1$ ), the probability to be found in the spin  $\pm\mathbf{e}$  state is given by  $(1 \pm \boldsymbol{\zeta} \cdot \mathbf{e})/2$ .

The polarization vector  $\boldsymbol{\zeta}$  obeys the Thomas-BMT equation (5.35) in the absence 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  $\mathbf{k}$ . It can be represented by the components along two unit vectors  $\mathbf{e}^{(1)}$  and  $\mathbf{e}^{(2)}$  perpendicular to  $\mathbf{k}$ . The three vectors ( $\mathbf{e}^{(1)}$ ,  $\mathbf{e}^{(2)}$ ,  $\mathbf{k}/|\mathbf{k}|$ ) form a right-handed orthonormal basis. The density matrix is defined as

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

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

$$\rho^{(\gamma)} = \frac{1}{2}(1 + \boldsymbol{\xi} \cdot \boldsymbol{\sigma}), \quad \boldsymbol{\xi} = \text{Trace}(\rho^{(\gamma)} \boldsymbol{\sigma}). \quad (5.7)$$

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

- $\xi_1$  Linear polarization along the direction  $(\mathbf{e}^{(1)} + \mathbf{e}^{(2)})/\sqrt{2}$  ( $\xi_1 > 0$ ) or  $(\mathbf{e}^{(1)} - \mathbf{e}^{(2)})/\sqrt{2}$  ( $\xi_1 < 0$ )
- $\xi_2$  Circular polarization
- $\xi_3$  Linear polarization along the direction  $\mathbf{e}^{(1)}$  ( $\xi_3 > 0$ ) or  $\mathbf{e}^{(2)}$  ( $\xi_3 < 0$ ).

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  $\xi_L$  is the magnitude of linear polarization and  $\phi_L$  (modulo  $\pi$ ) is the angle of the polarization plane measured from the  $\mathbf{e}^{(1)}$ -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  $\xi_i$ : one has to define the  $\mathbf{e}^{(1)}$ -axis. The most general way is that every macro-photon carries its own  $\mathbf{e}^{(1)}$ -axis but this is too much redundant. **CAIN** adopts the convention that  $\mathbf{e}^{(1)}$  is parallel to  $\mathbf{e}_x - (\mathbf{e}_x \cdot \mathbf{k})\mathbf{k}/|\mathbf{k}|^2$  where  $\mathbf{k}$  is the photon momentum. This is ill-defined when  $\mathbf{k}$  is parallel to  $\mathbf{e}_x$  but it will not cause a serious problem. (For lasers  $\mathbf{e}^{(1)}$ -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

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

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

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

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

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

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

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

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

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  $\zeta_i$  but this is not correct because of the selection effect due to the term  $\underline{\mathbf{f}} \cdot \zeta_i$ .

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

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

$$\sum_{\alpha} \sum_{\pm} \pm \frac{1 \pm \mathbf{e} \cdot \zeta_{i,\alpha}}{2} [1 - (\underline{w} \pm \underline{\mathbf{f}} \cdot \mathbf{e}) \Delta t] = \sum_{\alpha} [\mathbf{e} \cdot \zeta_{i,\alpha} (1 - \underline{w} \Delta t) - \underline{\mathbf{f}} \cdot \mathbf{e} \Delta t] = N \mathbf{e} \cdot [\zeta_i (1 - \underline{w} \Delta t) - \underline{\mathbf{f}} \Delta t].$$

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

$$\zeta_{f,\text{no trans.}} = \frac{\zeta_i (1 - \underline{w} \Delta t) - \underline{\mathbf{f}} \Delta t}{1 - (\underline{w} + \underline{\mathbf{f}} \cdot \zeta_i) \Delta t} \quad (5.12)$$

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

$$\begin{aligned} \zeta_f &= [1 - (\underline{w} + \underline{\mathbf{f}} \cdot \zeta_i) \Delta t] \zeta_{f,\text{notrans.}} + \int d\Gamma [w(\Gamma) + \underline{\mathbf{f}}(\Gamma) \cdot \zeta_i] \Delta t \zeta_{f,\text{trans.}} \\ &= \zeta_i + [\underline{\mathbf{g}} - \underline{\mathbf{f}} - \underline{w} \zeta_i + \underline{H} \zeta_i] \Delta t. \end{aligned} \quad (5.13)$$

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

$$\frac{d\zeta}{dt} = \underline{\mathbf{g}} - \underline{\mathbf{f}} - \underline{w} \zeta + \underline{H} \zeta. \quad (5.14)$$

### Polarization effects included in CAIN

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 <sup>±</sup>	laser	final e <sup>±</sup>	final γ
Beamstrahlung	e <sup>±</sup> → e <sup>±</sup> + γ	LT	–	LT	LT
Linear laser-Compton	e <sup>±</sup> + laser → e <sup>±</sup> + γ	LT	LT	LT	LT
Nonlinear laser-Compton	e <sup>±</sup> + n·laser → e <sup>±</sup> + γ or	L	L*	L	L
		N	T*	N	T
		initial γ	laser	final e <sup>±</sup>	
Coherent pair	γ → e <sup>+</sup> + e <sup>-</sup>	LT	–	LT	
Linear laser-Breit-Wheeler	γ + laser → e <sup>+</sup> + e <sup>-</sup>	LT	LT	LT	
Nonlinear laser-Breit-Wheeler	γ + n·laser → e <sup>+</sup> + e <sup>-</sup>	L	L*	L	
		initial	final pair		
Incoherent Breit-Wheeler	γ + γ → e <sup>+</sup> + e <sup>-</sup>	L	N		
Incoherent Bethe-Heitler	γ + e → e + e <sup>+</sup> + e <sup>-</sup>	N	N		
Incoherent Landau-Lifshitz	e + e → e + e + e <sup>+</sup> + e <sup>-</sup>	N	N		
		initial	final		
Bremsstrahlung	e + e → e + e + γ	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  $s=\text{constant}$  rather than  $t=\text{constant}$ . 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)
- $\sigma_t$  R.m.s. bunch length. (m)
- $\sigma_\epsilon$  R.m.s. relative energy spread.
- $n_x, n_y, n_t, n_\epsilon$  Gaussian tail cut off.
- $\theta_x, \theta_y$  Orbit slope. (rad)
- $\psi_x, \psi_y$  Crab angle. (rad)
- $\phi_{xy}$  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 using energy slope.)
$d\alpha_{xy}/dt$	Bunch position dependence of the focal point. (Parameter for the travelling focus using crab cavity and sextupole magnet.)

First, generate particle variables in usual accelerator coordinate:

$$t_1 = \sigma_t r_1 \quad (5.15)$$

$$\varepsilon_1 = \sigma_\varepsilon r_2 + (d\varepsilon/dt)t_1 \quad (5.16)$$

$$x_1 = \sqrt{2\varepsilon_x u_1 \beta_{x1}} \cos \varphi_1 + \eta_x \varepsilon_1 \quad (5.17)$$

$$x'_1 = \sqrt{2\varepsilon_x u_1 / \beta_{x1}} (-\alpha_{x1} \cos \varphi_1 - \sin \varphi_1) + \eta'_x \varepsilon_1 \quad (5.18)$$

$$y_1 = \sqrt{2\varepsilon_y u_2 \beta_{y1}} \cos \varphi_2 + \eta_y \varepsilon_1 \quad (5.19)$$

$$y'_1 = \sqrt{2\varepsilon_y u_2 / \beta_{y1}} (-\alpha_{y1} \cos \varphi_2 - \sin \varphi_2) + \eta'_y \varepsilon_1 \quad (5.20)$$

$$\alpha_{xy1} = \alpha_{xy} + \varepsilon_1 (d\alpha_{xy}/d\varepsilon) + t_1 (d\alpha_{xy}/dt), \quad (5.21)$$

$$\beta_{xy1} = \beta_{xy} [1 + (\varepsilon_1 d\alpha_{xy}/d\varepsilon + t_1 d\alpha_{xy}/dt)^2] \quad (5.22)$$

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

These variables are then transformed to

$$t = t_0 + t_1 \quad (5.23)$$

$$\begin{pmatrix} x \\ y \end{pmatrix} = \begin{pmatrix} x_0 \\ y_0 \end{pmatrix} + \begin{pmatrix} \psi_x \\ \psi_y \end{pmatrix} t_1 + \begin{pmatrix} \cos \phi_{xy} & -\sin \phi_{xy} \\ \sin \phi_{xy} & \cos \phi_{xy} \end{pmatrix} \begin{pmatrix} x_1 \\ y_1 \end{pmatrix} \quad (5.24)$$

$$\begin{pmatrix} x' \\ y' \end{pmatrix} = \begin{pmatrix} \theta_x \\ \theta_y \end{pmatrix} + \begin{pmatrix} \cos \phi_{xy} & -\sin \phi_{xy} \\ \sin \phi_{xy} & \cos \phi_{xy} \end{pmatrix} \begin{pmatrix} x'_1 \\ y'_1 \end{pmatrix} \quad (5.25)$$

$$s = s_0 \quad (5.26)$$

Finally, the energy-momentum is given by

$$E = E_0 + E_0 \varepsilon_1 \quad (5.27)$$

$$p_s = \pm \sqrt{\frac{E^2 - m^2}{1 + x'^2 + y'^2}} \quad (5.28)$$

$$p_x = |p_s| x' \quad (5.29)$$

$$p_y = |p_s| y' \quad (5.30)$$

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

$$\frac{dx^\mu}{d\tau} = \frac{1}{m}p^\mu, \quad \frac{dp^\mu}{d\tau} = \frac{e}{m}F^\mu{}_\nu p^\nu. \quad (5.31)$$

where  $\tau$  is the proper time and  $F^\mu{}_\nu$  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

$$\omega_1 = \sqrt{\frac{1}{2}(\sqrt{a^2 + 4b^2} + a)}, \quad \omega_2 = i\sqrt{\frac{1}{2}(\sqrt{a^2 + 4b^2} - a)} \quad (5.32)$$

with

$$a = \frac{e^2}{m^2}(\mathbf{E}^2 - \mathbf{B}^2), \quad b = \frac{e^2}{m^2}(\mathbf{E} \cdot \mathbf{B}). \quad (5.33)$$

Then, the solution is

$$p^\mu(\tau) = \sum_{\omega=\omega_1, \omega_2} \frac{\pm 1}{\sqrt{a^2 + 4b^2}} \left[ (\omega^2 + \tilde{f}^\mu{}_\alpha \tilde{f}^\alpha{}_\nu) \cosh \omega\tau + (\omega^2 f^\mu{}_\nu + b \tilde{f}^\mu{}_\nu) \frac{\sinh \omega\tau}{\omega} \right] p^\nu(0), \quad (5.34)$$

where the upper (lower) sign applies to  $\omega_1$  ( $\omega_2$ ) 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

$$\frac{d\mathbf{S}}{dt} = -\frac{e}{m\gamma} \left[ (\gamma a + 1)\mathbf{B}_T + (a + 1)\mathbf{B}_L - \gamma \left( a + \frac{1}{\gamma + 1} \right) \beta \mathbf{e}_v \times \frac{\mathbf{E}}{c} \right] \times \mathbf{S}, \quad (5.35)$$

where  $a$  is the coefficient of anomalous magnetic moment and

$$\mathbf{B}_L = \mathbf{e}_v(\mathbf{B} \cdot \mathbf{e}_v), \quad \mathbf{B}_T = \mathbf{B} - \mathbf{B}_L = \mathbf{e}_v \times (\mathbf{B} \times \mathbf{e}_v). \quad (5.36)$$

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  $\Upsilon$ .

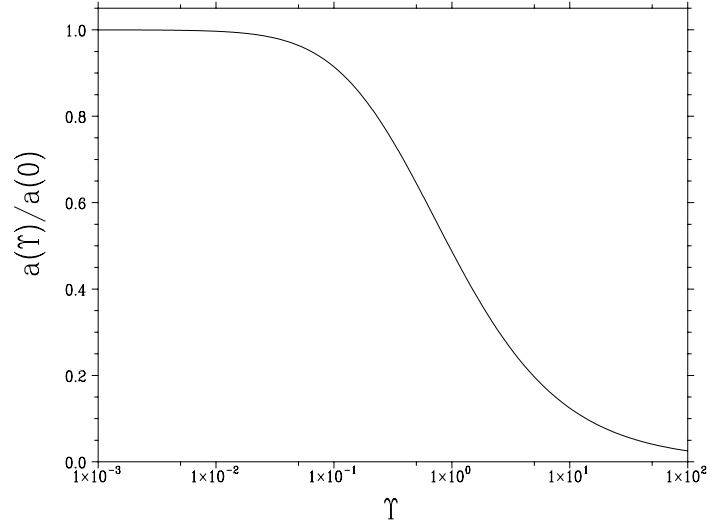
$$\Upsilon = \frac{e}{m^3} \sqrt{-(F^{\mu\nu} p_\nu)^2} = \frac{e}{m^3} \sqrt{(p^0 \mathbf{E}_T + \mathbf{p} \times \mathbf{B})^2 + \mathbf{E}_L^2}. \quad (5.37)$$

According to V. N. Baier (V.N. Baier, V.M. Katkov and V.M. Strakhovenko, Lett. Nuovo Cimento vol.15 issue 5 p149 (1976)),

$$\frac{a(\Upsilon)}{a(0)} = \frac{2}{\Upsilon} \int_0^\infty \frac{x dx}{(1+x)^3} \int_0^\infty \sin \left[ \frac{x}{\Upsilon} \left( t + \frac{t^3}{3} \right) \right] dt \quad (5.38)$$

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 presence of the beam field. The equation of motion is in general written in the form

$$\frac{d\mathbf{r}}{dt} = \mathbf{v}(\mathbf{p}) = \frac{\mathbf{p}}{\sqrt{m^2 + \mathbf{p}^2}} \quad (5.39)$$

$$\frac{d\mathbf{p}}{dt} = \mathbf{F}(\mathbf{r}, \mathbf{p}) \quad (5.40)$$

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

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

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

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

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

If this is not small enough, divide the interval  $\Delta t$  by an integer  $n_d$ . Note that  $\delta \mathbf{r} \propto (\Delta t)^3$  because  $\mathbf{F}_1 - \mathbf{F}_0$  is proportional to  $\Delta t$ . 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 becomes 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  $\Delta t$ , 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

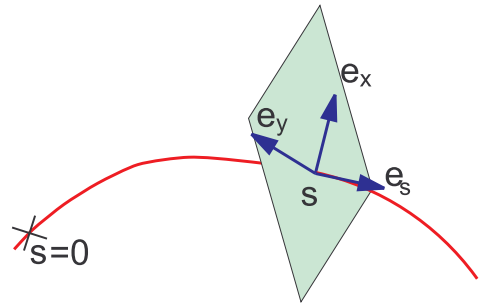
A beamline has its own coordinate system  $(t, x, y, s)$  which is in general a curvilinear coordinate, different from the Cartesian coordinate of CAIN. (At present the time coordinate is the same.)

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

$$\frac{d\mathbf{r}_0}{ds} = \mathbf{e}_s(s) \quad (5.41)$$

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

$$\frac{d\mathbf{e}_s}{ds} = \boldsymbol{\Omega}(s) \times \mathbf{e}_s, \quad \boldsymbol{\Omega}(s) \cdot \mathbf{e}_s = 0 \quad (5.42)$$



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

$$\frac{d\mathbf{e}_j}{ds} = \boldsymbol{\Omega}(s) \times \mathbf{e}_j \quad (j = x, y, s) \quad (5.43)$$

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

$$\boldsymbol{\Omega} = -\frac{\mathbf{e}_y(s)}{\rho_x(s)} + \frac{\mathbf{e}_x(s)}{\rho_y(s)}. \quad (5.44)$$

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

$$\mathbf{r} = \mathbf{r}_0(s) + x\mathbf{e}_x(s) + y\mathbf{e}_y(s) \quad (5.45)$$

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

$$\boldsymbol{\Omega}(s) = \frac{1}{\rho}(\mathbf{e}_x \sin \phi - \mathbf{e}_y \cos \phi), \quad \rho = \frac{l}{\theta} \quad (5.46)$$

By solving eqs.(5.41) and (5.43) with eq.(5.46) one finds the transportation of the basis vectors and  $\mathbf{r}_0(s)$  through the bending magnet as

$$\begin{pmatrix} \mathbf{e}_x \\ \mathbf{e}_y \\ \mathbf{e}_s \end{pmatrix}_{s_0+l} = \begin{pmatrix} \cos \phi & -\sin \phi & 0 \\ \sin \phi & \cos \phi & 0 \\ 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} \cos \theta & 0 & \sin \theta \\ 0 & 1 & 0 \\ -\sin \theta & 0 & \cos \theta \end{pmatrix} \begin{pmatrix} \cos \phi & \sin \phi & 0 \\ -\sin \phi & \cos \phi & 0 \\ 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} \mathbf{e}_x \\ \mathbf{e}_y \\ \mathbf{e}_s \end{pmatrix}_{s_0} \quad (5.47)$$

$$\mathbf{r}_0(s_0 + l) = \mathbf{r}_0(s_0) + \rho[\mathbf{e}_s(s_0) \sin \theta + (1 - \cos \theta)(\mathbf{e}_x(s_0) \cos \phi + \mathbf{e}_y(s_0) \sin \phi)] \quad (5.48)$$

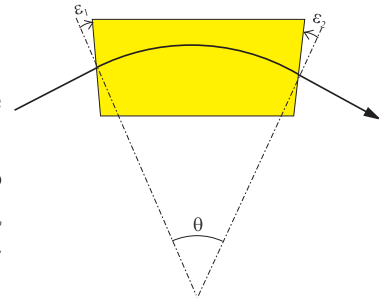
where  $s_0$  is the entrance of the magnet.

To locate the beamline in the **CAIN** coordinate system, we need  $\mathbf{r}_0$  and  $(\mathbf{e}_x, \mathbf{e}_y, \mathbf{e}_s)$  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  $k_1 = -\tan \epsilon / \rho_0$ ,  $\epsilon$ :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  $\theta_0$  (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

$$\alpha \equiv \tan^{-1}(p_x/p_s), \quad p_1 \equiv \sqrt{p_s^2 + p_x^2}, \quad \rho \equiv \pm \rho_0 \frac{p_1}{p_0}, \quad \omega_1 \equiv \frac{p_1}{\rho E},$$

$$f \equiv \sin(\alpha + \theta_0) - \frac{x + \rho_0}{\rho} \sin \theta_0 = 2 \sin \frac{\alpha}{2} \cos(\frac{\alpha}{2} + \theta_0) + \frac{\rho - \rho_0 - x}{\rho} \sin \theta_0$$

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

$$t_f = t + \Delta t, \quad \Delta t = \frac{1}{\omega_1}(\alpha + \theta_0 - \sin^{-1} f)$$

$$x_f = x \cos \theta_0 - \rho_0(1 - \cos \theta_0) + \rho(\sqrt{1 - f^2} - \cos(\alpha + \theta_0))$$

$$= x \cos \theta_0 + 2(\rho - \rho_0) \sin^2 \frac{\alpha + \theta_0}{2} + 2\rho_0 \sin \frac{\alpha}{2} \sin(\frac{\alpha}{2} + \theta_0) - \rho \frac{f^2}{1 + \sqrt{1 - f^2}}$$

$$y_f = y + \frac{p_y}{E} \Delta t$$

$$s_f = s + l$$

$$E_f = E$$

$$p_{xf} = f p_1$$

$$p_{yf} = p_y$$

$$p_{sf} = \sqrt{1 - f^2} p_1 > 0$$

### BMT equation

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

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

to be compared with the equation for momentum

$$\frac{d\mathbf{p}}{dt} = \left( \frac{-e\mathbf{B}}{m\gamma} \right) \times \mathbf{p} = -\omega \mathbf{e}_y \times \mathbf{p}, \quad \omega \equiv \frac{em_0\gamma_0 v_0}{e_0 m \gamma \rho_0}.$$

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

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

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

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

## 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

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

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  $\tilde{n}_j(\mathbf{r}, \mathbf{v}, t)$ . It is normalized such that  $\int \tilde{n}_j d\mathbf{r} d\mathbf{v}$  is the total number of particles in the  $j$ -th beam. The luminosity (per crossing) is in general given by

$$\mathcal{L} = \int \sqrt{(\mathbf{v}_1 - \mathbf{v}_2)^2 - (\mathbf{v}_1 \times \mathbf{v}_2)^2} \tilde{n}_1(\mathbf{r}, \mathbf{v}_1, t) \tilde{n}_2(\mathbf{r}, \mathbf{v}_2, t) d\mathbf{r} d\mathbf{v}_1 d\mathbf{v}_2 dt. \quad (5.49)$$

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

$$\mathcal{L} = (1 - \cos \phi) \int n_1(\mathbf{r}, t) n_2(\mathbf{r}, t) d\mathbf{r} dt \quad (5.50)$$

where  $\phi$  is the polar angle between  $\mathbf{v}_1$  and  $\mathbf{v}_2$ , and  $n_j(\mathbf{r}, t) = \int \tilde{n}_j d\mathbf{v}$  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  $\Delta t$ , longitudinal slice width  $\Delta_s$ , transverse mesh size  $\Delta_x$  and  $\Delta_y$ . Summing the number of particles in each bin, the luminosity is given by

$$\mathcal{L} = C \sum_{i_x, i_y, i_s, i_t} N_{i_x, i_y, i_s, i_t}^{(1)} N_{i_x, i_y, i_s, i_t}^{(2)} \Delta_x \Delta_y \Delta_s \Delta_t \quad (5.51)$$

where  $C$  is an appropriate normalization factor, and  $N_{i_x, i_y, i_s, i_t}^{(j)}$  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 ( $\Delta_x$  and  $\Delta_y$  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  $N_k^{(j)}$  ( $k = 0, 1, 2, \dots, n^2 - 1$ ).

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

$$N_{4k}^n + N_{4k+1}^n + N_{4k+2}^n + N_{4k+3}^n = N_k^{n/2} \quad (5.52)$$

where  $N_k^n$  is the number of particles in the  $k$ -th bin ( $k = 0, 1, 2, \dots, n^2 - 1$ ) 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  $(\xi_1, \xi_2, \xi_3)$  for photons or the polarization vector  $(\zeta_x, \zeta_y, \zeta_s)$  for electrons/positrons. Let us denote the three numbers in general by  $\mathbf{s} \equiv (s_1, s_2, s_3)$ .

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

$$\sigma = \sum_{i,j=0}^3 \sigma_{i,j} s_i^{(R)} s_j^{(L)}, \quad (5.53)$$

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.(5.53) with an appropriate factor over the momentum, the interaction volume and time:

$$N = \int d\mathbf{r} dt d\mathbf{p}^{(R)} d\mathbf{p}^{(L)} \sigma f \quad (5.54)$$

where  $f$  is the particle density functions with kinematic factors and is found in eq.(5.49). Since  $\mathbf{s}^{(R)}$  and  $\mathbf{s}^{(L)}$  depend on the particles in general, we get different coefficients from term to term of eq.(5.53). Thus, the number of events is written in the form

$$N = \sum_{i,j=0}^3 \sigma_{i,j} \mathcal{L}_{i,j}, \quad \mathcal{L}_{i,j} = \int d\mathbf{r} dt d\mathbf{p}^{(R)} d\mathbf{p}^{(L)} s_i^{(R)} s_j^{(L)} f \quad (5.55)$$

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  $\sigma$  is given by eq.(5.55) with beam polarization value  $s_i^{(R,L)}$  plugged-in.

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

$$\begin{aligned} \sigma_{++} &= \sigma_{00} + \sigma_{30} - \sigma_{03} - \sigma_{33} \\ \sigma_{-+} &= \sigma_{00} - \sigma_{30} - \sigma_{03} + \sigma_{33} \\ \sigma_{+-} &= \sigma_{00} + \sigma_{30} + \sigma_{03} + \sigma_{33} \\ \sigma_{--} &= \sigma_{00} - \sigma_{30} + \sigma_{03} - \sigma_{33} \end{aligned} \quad (5.56)$$

The number of events is written as

$$N = \sigma_{++} \mathcal{L}_{++} + \sigma_{-+} \mathcal{L}_{-+} + \sigma_{+-} \mathcal{L}_{+-} + \sigma_{--} \mathcal{L}_{--} \quad (5.57)$$

with

$$\begin{aligned} \mathcal{L}_{++} &= \frac{1}{4} (\mathcal{L}_{00} + \mathcal{L}_{30} - \mathcal{L}_{03} - \mathcal{L}_{33}) \\ \mathcal{L}_{-+} &= \frac{1}{4} (\mathcal{L}_{00} - \mathcal{L}_{30} - \mathcal{L}_{03} + \mathcal{L}_{33}) \\ \mathcal{L}_{+-} &= \frac{1}{4} (\mathcal{L}_{00} + \mathcal{L}_{30} + \mathcal{L}_{03} + \mathcal{L}_{33}) \\ \mathcal{L}_{--} &= \frac{1}{4} (\mathcal{L}_{00} - \mathcal{L}_{30} + \mathcal{L}_{03} - \mathcal{L}_{33}) \end{aligned} \quad (5.58)$$

The total luminosity is  $\mathcal{L}_{00}$ . Note that the helicity is  $\xi_2$  for photons. Thus, if both beams are photons, the above expression becomes

$$\begin{aligned} \mathcal{L}_{++} &= \frac{1}{4} (\mathcal{L}_{00} + \mathcal{L}_{20} + \mathcal{L}_{02} + \mathcal{L}_{22}) \\ \mathcal{L}_{-+} &= \frac{1}{4} (\mathcal{L}_{00} - \mathcal{L}_{20} + \mathcal{L}_{02} - \mathcal{L}_{22}) \\ \mathcal{L}_{+-} &= \frac{1}{4} (\mathcal{L}_{00} + \mathcal{L}_{20} - \mathcal{L}_{02} - \mathcal{L}_{22}) \\ \mathcal{L}_{--} &= \frac{1}{4} (\mathcal{L}_{00} + \mathcal{L}_{20} + \mathcal{L}_{02} + \mathcal{L}_{22}) \end{aligned} \quad (5.59)$$

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 (5.58) is not exact because the helicity is defined as  $\boldsymbol{\zeta} \cdot \mathbf{p} / |\mathbf{p}|$  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  $\mathbf{E}$ , the magnetic field is given by  $\mathbf{B} = \pm c\mathbf{e}_s \times \mathbf{E}$ , where  $\mathbf{e}_s$  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)-going 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  $\Delta s$  is defined by the parameter `Smesh`). Within each slice the following Poisson equation is solved.

$$\Delta\Phi(x, y) = 2\pi\rho_Q(x, y), \quad \mathbf{E} = \frac{2mr_e}{\Delta s}\nabla\Phi, \quad (5.60)$$

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  $\mathbf{E}$  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.(5.60) can formally be solved as

$$\Phi(\mathbf{r}) = \int_{-\infty}^{\infty} G(\mathbf{r} - \mathbf{r}')\rho_Q(\mathbf{r}')d\mathbf{r}', \quad G(\mathbf{r}) = \log|\mathbf{r}|. \quad (5.61)$$

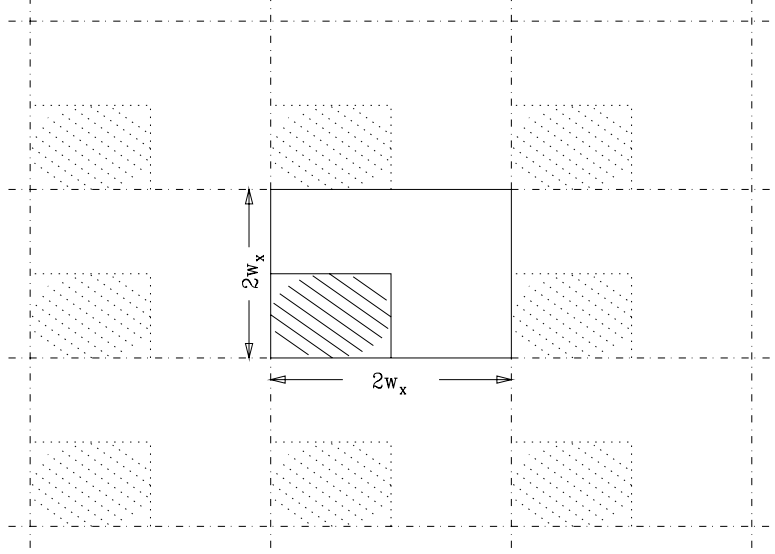
Divide this region by  $n_x \times n_y$  grid. Within each cell  $(i, j)$  ( $i = 1, \dots, n_x, j = 1, \dots, n_y$ ), the density  $\rho_Q(x, y)$  is approximated by  $Q_{ij}/\Delta_x\Delta_y$ , where  $\Delta_x = w_x/n_x$ ,  $\Delta_y = w_y/n_y$ , and  $Q_{ij}$  is the total charge in the cell:

$$Q_{ij} = \int_{(x,y) \text{ in mesh } (ij)} \rho_Q(x, y)dx dy. \quad (5.62)$$

where  $(x_i, y_j)$  is the cell center coordinate. Then, eq.(5.61) becomes a sum over the cells.

$$\Phi(x_i, y_j) = \sum_{i', j'} G_{i-i', j-j'} Q_{i', j'}. \quad (5.63)$$

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:

$$G_{i,j} = \frac{1}{\Delta_x \Delta_y} \int_{(x,y) \text{ in cell } (i,j)} \frac{1}{2} \log[(x_i - x)^2 + (y_j - y)^2] dx dy \quad (5.64)$$

This averaging is important when  $\Delta_x/\Delta_y$  is far from unity. The convolution in eq.(5.63) 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.(5.61), 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  $\Phi$  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 **CAIN2.44b**. 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, **CAIN2.44b** 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 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.

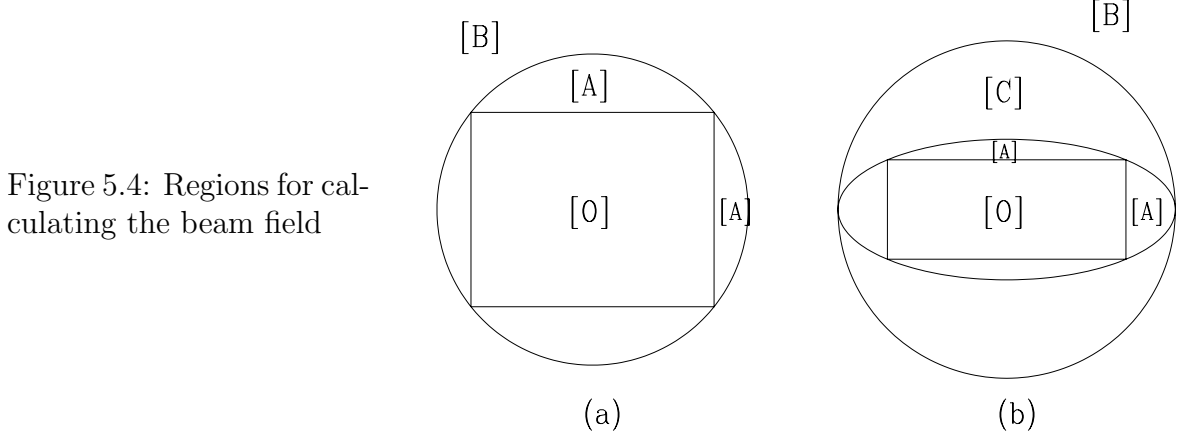


Figure 5.4: Regions for calculating the beam field

#### [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

$$\frac{\partial\Phi}{\partial x} + i\frac{\partial\Phi}{\partial y} = -\sum_{i,j} \frac{(x-x_i) + i(y-y_j)}{(x-x_i)^2 + (y-y_j)^2} Q_{ij}, \quad (5.65)$$

However, this formula is not accurate when the bin size ratio  $\Delta_x/\Delta_y$  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  $(\Delta_x, \Delta_y)$  for faster computation.

#### [B] Harmonic expansion in polar coordinate

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

$$\frac{\partial\Phi}{\partial x} - i\frac{\partial\Phi}{\partial y} = -\sum_{m=0}^{\infty} \left[ \frac{r_0}{x+iy} \right]^{m+1} Q_m^B, \quad (5.66)$$

$$Q_m^B = \frac{1}{r_0} \int \rho_Q(x, y) \left[ \frac{x+iy}{r_0} \right]^m dx dy. \quad (5.67)$$

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

$$\begin{aligned} x &= f \cosh u \cos v & \cosh(u + iv) &= (x + iy)/f \\ y &= f \sinh u \sin v & (u \geq 0, \quad 0 \leq v < 2\pi) & \end{aligned} \quad (5.68)$$

is used. Here,  $f$  is chosen as

$$f = \sqrt{(w_x^2 - w_y^2)/2}. \quad (5.69)$$

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

$$u_0 = \frac{1}{2} \log \frac{w_x + w_y}{w_x - w_y}, \quad (5.70)$$

which is taken at the four corners.

Then, the expansion of  $\Phi$  is

$$\frac{\partial \Phi}{\partial x} - i \frac{\partial \Phi}{\partial y} = - \sum_{m=0}^{\infty} e^{-(m+1)(u-u_0+iv)} Q_m^c, \quad (5.71)$$

$$Q_m^c = \frac{2}{f} e^{-(m+1)u_0} \int \rho_Q(x, y) \frac{\sinh[(m+1)(u+iv)]}{\sinh(u+iv)} dx dy. \quad (5.72)$$

Actually, there is a finite relation between  $Q_m^B$  and  $Q_m^c$ :

$$Q_m^c = e^{-(m+1)u_0} \sum_{r=0}^{[m/2]} (-1)^r \binom{m-r}{r} \left( \frac{2r_0}{f} \right)^{m-2r+1} Q_{m-2r}^B. \quad (5.73)$$

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

#### 5.8.1.1 Definition of laser coordinate

Define a coordinate system attached to a laser. Let  $\mathbf{e}^{(3)}$  be the unit vector along the direction of propagation, and introduce a unit vector  $\mathbf{e}^{(1)}$  perpendicular to  $\mathbf{e}^{(3)}$  and another unit vector  $\mathbf{e}^{(2)} = \mathbf{e}^{(3)} \times \mathbf{e}^{(1)}$ . The three vectors  $(\mathbf{e}^{(1)}, \mathbf{e}^{(2)}, \mathbf{e}^{(3)})$  form an orthonormal frame. Define the components of these vectors in the original frame  $(\mathbf{e}_x, \mathbf{e}_y, \mathbf{e}_s)$  as

$$\mathbf{e}^{(1)} = \begin{pmatrix} V_{11} \\ V_{21} \\ V_{31} \end{pmatrix}, \quad \mathbf{e}^{(2)} = \begin{pmatrix} V_{12} \\ V_{22} \\ V_{32} \end{pmatrix}, \quad \mathbf{e}^{(3)} = \begin{pmatrix} V_{13} \\ V_{23} \\ V_{33} \end{pmatrix}. \quad (5.74)$$

Then,  $V = \{V_{ij}\}$  is a  $3 \times 3$  orthogonal matrix. Let  $(\xi, \eta, \zeta)$  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  $\tau$  whose origin is  $t_0$ . Now, the relation between  $(t, x, y, s)$  and  $(\tau, \xi, \eta, \zeta)$  is

$$\tau = t - t_0 \quad (5.75)$$

$$\begin{pmatrix} \xi \\ \eta \\ \zeta \end{pmatrix} = \begin{pmatrix} V_{11} & V_{21} & V_{31} \\ V_{12} & V_{22} & V_{32} \\ V_{13} & V_{23} & V_{33} \end{pmatrix} \begin{pmatrix} x - x_0 \\ y - y_0 \\ s - s_0 \end{pmatrix}. \quad (5.76)$$

### 5.8.1.2 Gaussian laser beam

A plane wave is written in the form, in the  $(\tau, \xi, \eta, \zeta)$  coordinate,

$$\mathbf{E} = \Re \mathbf{E}_0 e^{ik(\mathbf{n} \cdot \mathbf{r} - \tau)} \quad (5.77)$$

where  $k = 1/\lambda_L = 2\pi/\lambda_L$  is the wave number,  $\mathbf{n}$  is the unit vector along the propagation direction of the wave component, and  $\mathbf{E}_0$  is a complex vector perpendicular to  $\mathbf{n}$ . A laser beam is considered to be a superposition of plane waves with slightly different  $\mathbf{n}$  and  $k$ . If the distribution of  $\mathbf{n}$  around  $\mathbf{e}^{(3)}$  and that of  $k$  are Gaussian and if one ignores the  $\mathbf{n}$  dependence of  $\mathbf{E}_0$ , the laser field can be approximated by

$$\mathbf{E} = \Re \mathbf{E}_0 e^{ik(\zeta - \tau)} \sqrt{F} e^{i\Phi}, \quad (5.78)$$

where

$$F(\tau, \xi, \eta, \zeta) = F_t(\tau - \zeta) F_s(\xi, \eta, \zeta) \quad (5.79)$$

$$F_t = \exp \left[ -\frac{(\zeta - \tau)^2}{2\sigma_\tau^2} \right] \quad (5.80)$$

$$F_s = \frac{1}{\sqrt{[1 + (\zeta/\beta_1)^2][1 + (\zeta/\beta_2)^2]}} \exp \left[ -\frac{\xi^2}{2\sigma_1(\zeta)^2} - \frac{\eta^2}{2\sigma_2(\zeta)^2} \right]. \quad (5.81)$$

with

$$\sigma_i(\zeta) = \sqrt{d_i \frac{\lambda_L}{4\pi} \left( \beta_i + \frac{\zeta^2}{\beta_i} \right)}, \quad (i = 1, 2). \quad (5.82)$$

Here  $\beta_i$  ( $i = 1, 2$ ) is the Rayleigh length and  $\sigma_\tau$  is the r.m.s. pulse length. The dilatation factor  $d_i$  is introduced for later use.<sup>1</sup>

The wave front is given by the contour of  $k\zeta + \Phi$ . If one defines  $\mathbf{n}'$  by  $k\mathbf{n}' = \nabla(k\zeta + \Phi)$ ,  $\mathbf{n}'$  is nearly a unit vector and approximated by

$$\mathbf{n}' = \frac{\mathbf{e}^{(3)} + c_1 \mathbf{e}^{(1)} + c_2 \mathbf{e}^{(2)}}{\sqrt{1 + c_1^2 + c_2^2}}, \quad (5.83)$$

$$c_1 = \frac{\xi\zeta}{d_1(\beta_1^2 + \zeta^2)}, \quad c_2 = \frac{\eta\zeta}{d_2(\beta_2^2 + \zeta^2)}. \quad (5.84)$$

<sup>1</sup> In earlier versions of the manual the function  $F_s$  ( $A_s$ ) used to include the factor  $1/\sqrt{d_1 d_2}$ . This was a mistake of the manual. The factor is included in the definition of  $P_{00}$  specified by the input data POWERDENSITY.

In **CAIN**, when the relevant particle is at  $(t, x, y, s)$ , or at  $(\tau, \xi, \eta, \zeta)$  in laser coordinate, the laser field is considered to be locally a plane wave with the power density  $A(\tau, \xi, \eta, \zeta)P_{00}$ , wave number  $k$ , and the propagation direction  $\mathbf{n}'(\xi, \eta, \zeta)$ .

There is some problem on the polarization because eq.(5.78) does not exactly satisfy the Maxwell equation. For simplicity, the basis  $(\mathbf{e}'^{(1)}, \mathbf{e}'^{(2)}, \mathbf{e}'^{(3)})$  ( $\mathbf{e}'^{(3)} = \mathbf{n}'$ ) for polarization is defined in the following manner:  $\mathbf{e}'^{(1)}$  is the unit vector along  $\mathbf{e}^{(1)} - (\mathbf{e}^{(1)} \cdot \mathbf{n}')\mathbf{n}'$  and  $\mathbf{e}'^{(2)} = \mathbf{n}' \times \mathbf{e}'^{(1)}$ . (This is irrelevant if only the longitudinal polarization is needed.)

The r.m.s. beam size at the position  $\zeta$  is given by eq.(5.82) where the dilatation factor  $d_i$  is included. 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 (5.78) to (5.84) apply to Gaussian beam. When the time (space) structure of the pulse is given by a file,  $F_t$  ( $F_s$ ) is replaced by the values read from the file.

The Lorentz transformation is a little complicated because eq.(5.78) is far from a co-variant 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  $(A, \omega', \mathbf{e}'^{(1)}, \mathbf{e}'^{(2)}, \mathbf{e}'^{(3)})$  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  $\mathbf{n}'$  as in eq.(5.84). Laser field is treated locally as a plane wave.

#### 5.8.1.3 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

$$F(r, \zeta) = \frac{k}{\zeta + f} \exp \left\{ i \frac{kr^2}{2(\zeta + f)} \right\} \int_b^a d\rho \sqrt{\rho(\rho - b)} \exp \left\{ -\frac{(\rho - b)^2}{4\sigma_0^2} - i \frac{k\rho^2\zeta}{2(\zeta + f)f} \right\} J_0 \left( \frac{kr\rho}{\zeta + f} \right)$$

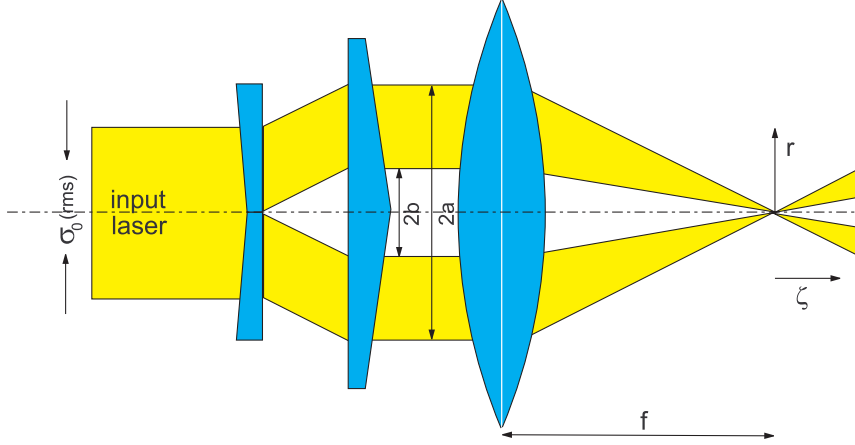


Figure 5.5: Geometry of axicon

Integration over the transverse plane  $\zeta=\text{const}$  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 ( $b = 0$ ,  $a = \infty$ ),

$$|F(r, \zeta)|^2 = \frac{\sigma_0^2}{\sigma(r, \zeta)^2} e^{-r^2/2\sigma(r, \zeta)^2}, \quad \sigma(r, \zeta) = \frac{\zeta + f}{2k\sigma_0} \sqrt{1 + \left(\frac{2k\sigma_0^2\zeta}{(\zeta + f)f}\right)^2}$$

$$\text{Rayleigh length} = \frac{f^2}{2k\sigma_0^2} = \frac{\lambda}{4\pi} \left(\frac{f}{\sigma_0}\right)^2$$

### 5.8.1.4 Laser field propagation in vaccuum

**CAIN** accepts the laser field profile wiriiten in a file (See Sec.3.6.4). In the special case **ORDER=XY**, instead of specifying 3-dimensional data at  $(\xi, \eta, \zeta)$ , one can specify only the 2-dimensional data at  $(\xi, \eta)$  on the plane  $\zeta = 0$ . Assuming the propagation of a perfectly coherent beam in free vaccuum, **CAIN** can extend the data to 3-dimensional space.

When the  $f$ -number is not too small ( $\gtrsim 1.5$ ), the longitudinal field  $E_\zeta$  can be ignored and the transverse field  $\mathbf{E}_\perp \approx \mathbf{E}$  varies only slowly ( $|\partial^2 \mathbf{E}/\partial \zeta^2| \ll |k\partial \mathbf{E}/\partial \zeta|$ ). Then the Maxwell equation of propagation in vaccuum is

$$\frac{\partial \mathbf{E}}{\partial \zeta} = \frac{i\lambda}{4\pi} \Delta_\perp \mathbf{E} = \frac{i\lambda}{4\pi} \left( \frac{\partial^2}{\partial \xi^2} + \frac{\partial^2}{\partial \eta^2} \right) \mathbf{E} \quad (5.85)$$

When  $\mathbf{E}(\xi, \eta, \zeta)$  is given on the plane  $\zeta = \zeta_0$ , the solution at  $\zeta \neq \zeta_0$  is given by

$$\hat{\mathbf{E}}(u, v; \zeta_0) = \int d\xi d\eta e^{-2\pi i(u\xi + v\eta)} \mathbf{E}(\xi, \eta; \zeta_0) \quad (5.86)$$

$$\mathbf{E}(\xi, \eta; \zeta) = \int du dv e^{2\pi i(u\xi + v\eta)} e^{-i\pi\lambda(\zeta - \zeta_0)(u^2 + v^2)} \hat{\mathbf{E}}(u, v; \zeta_0) \quad (5.87)$$

or by

$$\mathbf{E}(\xi, \eta; \zeta) = \frac{-i}{\lambda(\zeta - \zeta_0)} \int d\xi' d\eta' e^{\frac{i\pi}{\lambda(\zeta - \zeta_0)} [(\xi - \xi')^2 + (\eta - \eta')^2]} \mathbf{E}(\xi', \eta'; \zeta_0) \quad (5.88)$$

Both of these two expressions are exact solutions of eq.(5.85) but the former (latter) is preferred for short (long) distance propagation in numerical computation.

## 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:

- $\omega, \omega'$  Initial (laser) and final energies of the photon.
- $\mathbf{k}, \mathbf{k}'$  Initial (laser) and final momenta of the photon.
- $\theta, \phi$  Polar and azimuthal scattering angle of the photon.
- $d\Omega$  Solid angle =  $\sin\theta d\theta d\phi = (m/\omega'^2) d\omega' d\phi$ .
- $\xi^{(L)}, \xi'^{(L)}$  Photon Stokes parameters before and after collision as defined in[3](page 361).<sup>2</sup> The definition of the axis is such that the first axis is parallel to  $\mathbf{k} \times \mathbf{k}'$  and the third axis is  $\mathbf{k}$  ( $\mathbf{k}'$ ) for  $\xi^{(L)}$  ( $\xi'^{(L)}$ ).

The range of  $\omega'$  is given by

$$\frac{\omega}{1 + \lambda} \leq \omega' \leq \omega, \quad \lambda = \frac{2\omega}{m} \quad (5.89)$$

The Compton relation is

$$\frac{1}{\omega'} - \frac{1}{\omega} = \frac{1 - \cos\theta}{m}. \quad (5.90)$$

The crosssection is given by eq(87.22) in [3].<sup>3</sup>:

$$\begin{aligned} \frac{d\sigma}{d\Omega} = \frac{1}{8} r_e^2 \left( \frac{\omega'}{\omega} \right)^2 \times & \left[ F_0 + F_3(\xi_3^{(L)} + \bar{\xi}_3'^{(L)}) + F_{11}\xi_1^{(L)}\bar{\xi}_1'^{(L)} + F_{22}\xi_2^{(L)}\bar{\xi}_2'^{(L)} + F_{33}\xi_3^{(L)}\bar{\xi}_3'^{(L)} \right. \\ & \left. + (\mathbf{f} \cdot \boldsymbol{\zeta} + \mathbf{g} \cdot \bar{\boldsymbol{\zeta}}') \xi_2^{(L)} + (\mathbf{f}' \cdot \boldsymbol{\zeta} + \mathbf{g}' \cdot \bar{\boldsymbol{\zeta}}') \bar{\xi}_2'^{(L)} + \bar{\boldsymbol{\zeta}}' \cdot \mathcal{G} \cdot \boldsymbol{\zeta} + \dots \right] \quad (5.91) \end{aligned}$$

See Sec.5.2.2 for the meaning of the bars on  $\boldsymbol{\zeta}$  and  $\xi'^{(L)}$ . The omitted terms are products of three and four among  $\boldsymbol{\zeta}$ ,  $\bar{\boldsymbol{\zeta}}'$ ,  $\xi^{(L)}$ , and  $\bar{\xi}'^{(L)}$ . (Actually, we need the terms  $\boldsymbol{\zeta} \times \xi \times \bar{\boldsymbol{\zeta}}'$  and

<sup>2</sup> Actually, [3] adopts left-handed basis after collision so that the first and the second components of  $\xi'$  are  $-\xi'_1$  and  $-\xi'_2$  in [3]. Our  $\xi'^{(L)}$  is not  $(-\xi'_1, -\xi'_2, \xi'_3)$  but  $(\xi'_1, \xi'_2, \xi'_3)$ .

<sup>3</sup>The term  $\mathcal{G}$  is given in eq(4.6) in [4]



$\zeta \times \xi \times \bar{\xi}'$  but they are not found in literature.) The functions introduced in the above expression are:

$$F_0 = \frac{\omega}{\omega'} + \frac{\omega'}{\omega} - \sin^2 \theta, \quad F_3 = \sin^2 \theta, \quad (5.92)$$

$$F_{11} = 2 \cos \theta, \quad F_{22} = \left( \frac{\omega}{\omega'} + \frac{\omega'}{\omega} \right) \cos \theta, \quad F_{33} = 1 + \cos^2 \theta, \quad (5.93)$$

$$\mathbf{f} = -\frac{1}{m}(1 - \cos \theta)(\mathbf{k} \cos \theta + \mathbf{k}'), \quad \mathbf{f}' = -\frac{1}{m}(1 - \cos \theta)(\mathbf{k} + \mathbf{k}' \cos \theta), \quad (5.94)$$

$$\begin{bmatrix} \mathbf{g} \\ \mathbf{g}' \end{bmatrix} = \begin{bmatrix} \mathbf{f} \\ \mathbf{f}' \end{bmatrix} + \frac{\sin^2 \theta}{m} \frac{\omega + \omega'}{\omega - \omega' + 2m} (\mathbf{k} - \mathbf{k}'), \quad (5.95)$$

$$\begin{aligned} \mathcal{G} = & I(1 + \cos^2 \theta + \frac{\omega - \omega'}{2m} \sin^2 \theta) + \frac{\omega + \omega'}{2m}(1 + \cos \theta)(\mathbf{n}' \otimes \mathbf{n} - \mathbf{n} \otimes \mathbf{n}') \\ & - \frac{\omega - \omega'}{2m} [(\mathbf{n} + \mathbf{n}') \otimes (\mathbf{n} + \mathbf{n}') + (\mathbf{n} \times \mathbf{n}') \otimes (\mathbf{n} \times \mathbf{n}')] \\ & + \frac{\omega - \omega'}{m(\omega - \omega' + 2m)} (1 + \cos \theta)(\mathbf{k} - \mathbf{k}') \otimes (\mathbf{n} + \mathbf{n}') \\ & - \frac{\sin^2 \theta + 2 \cos \theta}{m(\omega - \omega' + 2m)} (\mathbf{k} - \mathbf{k}') \otimes (\mathbf{k} - \mathbf{k}'). \end{aligned} \quad (5.96)$$

These formulas are used in their exact forms in **CAIN**.

Summation over the final polarization and the azimuthal angle  $\phi$  gives the differential crosssection with respect to the final photon energy  $\omega'$ . Introducing the variables  $\bar{z}$  in place of  $\omega'$  by

$$\bar{z} = z/L_\lambda \quad (0 \leq \bar{z} \leq 1), \quad z = \log(\omega/\omega'), \quad L_\lambda \equiv \log(1 + \lambda), \quad (5.97)$$

$$\cos \theta = 1 - 2(e^z - 1)/\lambda, \quad (5.98)$$

we write the differential crosssection as

$$\frac{d\sigma}{d\bar{z}} = 4\pi r_e^2 \frac{L_\lambda}{\lambda} F(\bar{z}), \quad (5.99)$$

where

$$F(\bar{z}) = \frac{1}{2L_\lambda} \left[ 1 + e^{-2z} - e^{-z} \sin^2 \theta - h \cos \theta (1 - e^{-2z}) \right], \quad (5.100)$$

$$h = \frac{\mathbf{k}}{\omega} \cdot \boldsymbol{\zeta} \xi_2^{(L)}. \quad (5.101)$$

Note that  $\xi_3$  does not appear here because it is based on the scattering plane and, therefore, disappears after integration over the azimuthal angle. The function  $F(\bar{z})$  satisfies  $0 \leq F \leq 1$  for any  $\bar{z}$  and  $\lambda$  and is  $O(1)$  except when  $h$  is close to +1 and  $\lambda$  is extremely large. The Function  $F(\bar{z})$  is plotted in Fig.5.6.

The total crosssection for given initial momenta and polarizations is given by

$$\sigma_{tot} = 4\pi r_e^2 \frac{L_\lambda}{\lambda} F_{int}(\lambda), \quad (5.102)$$

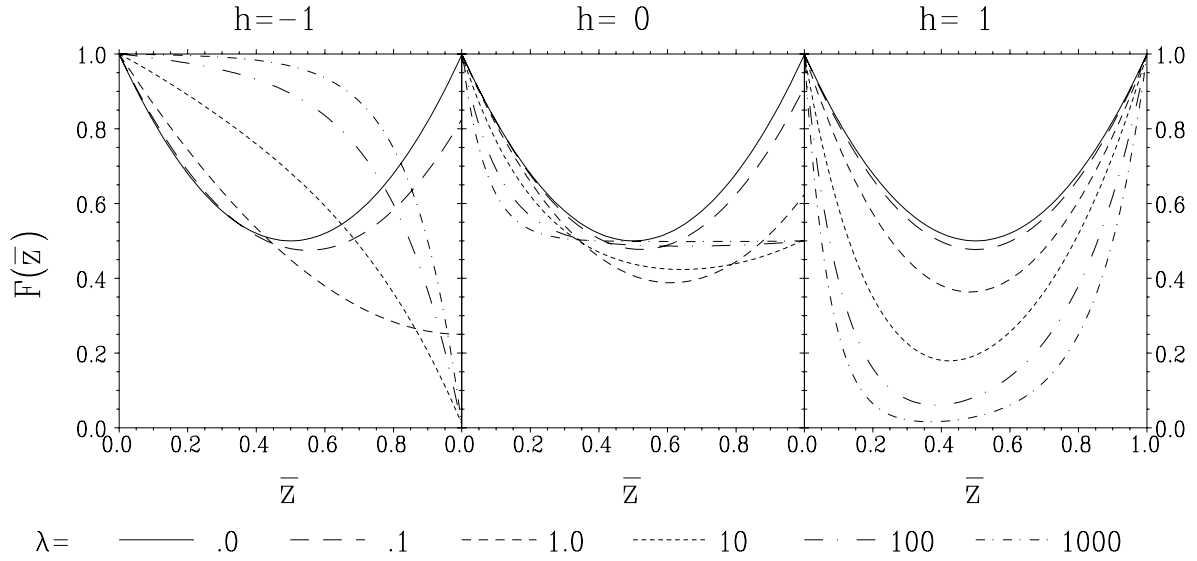
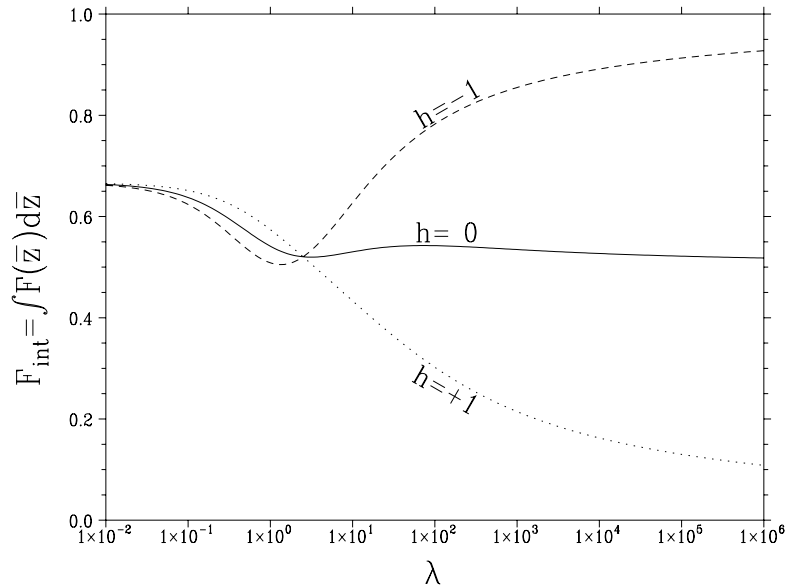


Figure 5.6: Function  $F(\bar{z})$  for  $h = 0, \pm 1$  for various values of  $\lambda$

$$F_{int}(\lambda) = \frac{1}{2L_\lambda} \left\{ \left(1 - \frac{4}{\lambda} - \frac{8}{\lambda^2}\right)L_\lambda + \frac{1}{2} + \frac{8}{\lambda} - \frac{1}{2(1+\lambda)^2} + h \left[ -\left(1 + \frac{2}{\lambda}\right)L_\lambda + 2 + \frac{1}{2(1+\lambda)^2} \right] \right\} \quad (5.103)$$

Figure 5.7: Function  $F_{int}(\lambda)$  for  $h = 0, \pm 1$  as a function of  $\lambda$ .  $F_{int}$  is less than unity and is  $O(1)$  unless  $h$  is close to  $+1$  and  $\lambda$  is extremely large.



Let us briefly describe the algorithm of event generation.

1. Compute the total event rate  $P_0$  in the given time interval using  $\sigma_{tot}$  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  $\xi_2$  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  $\bar{z}$  and  $r_2$  in  $(0,1)$ . Repeat this step until  $r_2 < F(\bar{z})$  is satisfied. Once or twice repetition is normally enough unless  $h$  is close to  $+1$  and  $\lambda$  is very large.
5. Compute  $\omega'$  from  $\bar{z}$ . 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 no longer be used. The laser field strength is characterized by the parameter

$$\xi = \frac{e\sqrt{-\langle A^\mu A_\mu \rangle}}{m} = \frac{\lambda_L}{m} \sqrt{\mu_0 c P} \quad (5.104)$$

Here,  $A^\mu$  is the 4-vector potential of the laser field and  $\langle \rangle$  indicates the average over the phase.  $\lambda_L$  is the laser wavelength  $/(2\pi)$  in meter,  $m$  the electron rest mass in  $\text{eV}/c^2$ ,  $c$  the velocity of light in  $\text{m/s}$ ,  $\mu_0 = 4\pi \times 10^{-7}$ , and  $P$  the power density in  $\text{Watt}/\text{m}^2$ . When  $\xi \ll 1$ , the well-known formulas are enough but as  $\xi$  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<sup>4</sup> 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 CAIN2.35. **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.
$\mathcal{E}, \omega_L, \mathcal{E}', \omega$	Energies of initial electron, laser photon, final electron and emitted photon, respectively.

---

<sup>4</sup>This should be treated by the CFQED command. If users want, EXTERNALFIELD command will be rewritten so as to accept varying fields.

$\lambda$  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  $p^\mu$  goes adiabatically into a strong laser field characterized by  $\xi$ , it behaves as if its momentum (called ‘quasi-momentum’) is

$$q = p + \frac{m^2 \xi^2}{2p \cdot k_L} k_L, \quad (q^2 = (1 + \xi^2)m^2). \quad (5.105)$$

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  $q'^\mu$  is defined like  $q^\mu$  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.

$\Theta$  Lorentz invariant variable defined by

$$\Theta \equiv \sqrt{n\lambda \frac{1-x}{x} - (1 + \xi^2)} = \sqrt{(1 + \xi^2) \left( \frac{v_n}{v} - 1 \right)} \quad (5.106)$$

The photon scattering angle  $\theta_\gamma$  ( $\theta_\gamma = 0$  for backward) in the head-on frame can be written as  $\theta_\gamma \approx m\Theta / \mathcal{E}$ .

Once  $x$  and  $n$  are given, the final momenta are given in the head-on frame by

$$k^\mu = xp^\mu + \left[ n(1-x) - \frac{2+\xi^2}{\lambda}x \right] k_L^\mu + mx\Theta (e_1^\mu \cos \phi + e_2^\mu \sin \phi) \quad (5.107)$$

$$p'^\mu = p^\mu + \left( n - \frac{\xi^2}{\lambda} \frac{x}{1-x} \right) k_L^\mu - k^\mu. \quad (5.108)$$

Here,  $e_1^\mu$  is a spatial vector perpendicular to  $\mathbf{p}$  (and to  $\mathbf{k}_L$ )<sup>5</sup> and  $e_2^\mu$  is defined by

$$e_2^\mu \equiv \frac{\epsilon_{\mu\nu\alpha\beta} e_1^\nu p^\alpha k_L^\beta}{p \cdot k_L} = \begin{bmatrix} 0 \\ \mathbf{e}_1 \times (\mathbf{p}\omega_L - \mathcal{E}\mathbf{k}_L) / p \cdot k_L \end{bmatrix}, \quad (5.109)$$

where  $\epsilon^{\mu\nu\alpha\beta}$  is the completely anti-symmetric tensor ( $\epsilon^{0123} = +1$ ).  $\phi$  is the azimuthal scattering angle in the head-on frame (its distribution is uniform in  $[0, 2\pi]$  for circularly polarized laser).

<sup>5</sup>The direction of  $e_1^\mu$  is arbitrary so long as perpendicular to  $\mathbf{p}$ . For convenience we choose the polarization plane of the laser for the case of linear polarization.

### 5.8.3.2 Case of Circularly Polarized Lasers

#### Transition rate

The transition rate per unit time is given by

$$W = \frac{\alpha m^2 \xi^2}{4q_0} \sum_{n=1}^{\infty} \int_0^{x_n} dx [(1+h_e \bar{h}_{e'}) F_{1n} + h_L (h_e + \bar{h}_{e'}) F_{2n} + h_e \bar{h}_{e'} F_{5n} + \bar{h}_\gamma (h_L F_{3n} + h_e F_{4n})] \quad (5.110)$$

$h_L$	Laser helicity ( $-1$ or $+1$ )
$h_e, h_{e'}$	Initial and final electron helicities ( $-1 \leq h_e \leq 1$ )
$h_\gamma$	Final photon helicity
$\bar{h}_{e'}, \bar{h}_\gamma$	‘Detector helicity’ of the final particles. See section 65 of [3].

The total transition rate is (sum over  $\bar{h}_{e'}$  and  $\bar{h}_\gamma$ )

$$W = \frac{\alpha m^2 \xi^2}{4q_0} \sum_{n=1}^{\infty} \int_0^{x_n} dx [F_{1n} + h_L h_e F_{2n}] \quad (5.111)$$

The functions  $F_{kn}$  are defined by

$$F_{1n} = -\frac{1}{\xi^2} J_n^2 + \frac{1}{2} \left[ 1 + \frac{v^2}{2(1+v)} \right] (J_{n-1}^2 + J_{n+1}^2 - 2J_n^2) \quad (5.112)$$

$$F_{2n} = \left( \frac{1}{2} - \frac{v}{v_n} \right) \frac{v(1+v/2)}{1+v} (J_{n-1}^2 - J_{n+1}^2) \quad (5.113)$$

$$F_{3n} = \left( \frac{1}{2} - \frac{v}{v_n} \right) \left[ 1 + \frac{v^2}{2(1+v)} \right] (J_{n-1}^2 - J_{n+1}^2) \quad (5.114)$$

$$F_{4n} = -\frac{v}{1+v} \frac{1}{\xi^2} J_n^2 + \frac{1}{2} \frac{v(1+v/2)}{1+v} (J_{n-1}^2 + J_{n+1}^2 - 2J_n^2) \quad (5.115)$$

$$F_{5n} = -\frac{v^2}{1+v} \frac{1}{\xi^2} J_n^2 \quad (5.116)$$

$J_n$ 's are Bessel functions with the argument  $z_n$  defined by

$$z_n = 2n \frac{\xi}{\sqrt{1+\xi^2}} \sqrt{\frac{v}{v_n} \left( 1 - \frac{v}{v_n} \right)} \quad (5.117)$$

$F_{1n}, F_{2n}, F_{3n}, F_{4n}$  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,  $\xi^2$ ,  $\lambda$ , and  $y = v/v_n$  ( $0 \leq y \leq 1$ ) are used as the independent variables instead of  $(\xi, \lambda, x)$ . The functions  $F_{kn}$  ( $k=1,2$ ) are stored in a 5-dimensional array **FF**( $k,n,i,j,l$ ) ( $n=1,\text{NPH}$ ), ( $i=0,\text{NY}$ ), ( $j=0,\text{NXI}$ ), ( $l=0,\text{NLAMBDA}$ ). (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

$0 \leq y \leq 1$  is then  $\text{FINT}(k,n,\text{NY},j,l)$  For integration, the trapezoidal rule is used, which means the function  $F_{kn}$  is approximated by a piecewise linear function. The sum of  $\text{FINT}(k,n,\text{NY},j,l)$  over  $n$  is stored in  $\text{FALL}(k,j,l)$ .

For a given initial condition, calculate the parameters  $\xi^2$  and  $\lambda$  and find  $\text{FALL}(\ast)$  by 2-dimensional interpolation. (The asterisk  $\ast$  indicates the appropriate sum over the initial polarization, *i.e.*,  $\text{FALL}(\ast)=\text{FALL}_{k=1}+h_L h_e \text{FALL}_{k=2}$ ). Then, calculate the total probability  $P$  (eq.(5.111) times the time interval  $\text{DT}$ ):

$$P = C_0 \times \text{FALL}(\ast), \quad C_0 = \frac{\alpha m^2 \xi^2 \Delta t}{\mathcal{E}_{eff}}. \quad (5.118)$$

If this is larger than the value defined by  $\text{PMAX}$  operand of  $\text{LASERQED}$  command, the time interval is subdivided such that  $P$  becomes less than  $\text{PMAX}$ .

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

$$h_{e, \text{new}} = \frac{h_e(1 - P_1) - h_L P_2}{1 - P}, \quad P_k = C_0 \times \text{FALL}_k \quad (k = 1, 2). \quad (5.119)$$

If accepted, decide how many laser photons to absorb. To do so, sum up  $\text{FINT}(\ast,n,\text{NY},j,l)$  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

$$\int_0^y dy \text{FF}(\ast) = r_2 \times \text{FINT}(\ast)$$

where  $r_2$  is another uniform random number. The left hand side is known for the mesh point of  $y$  (*i.e.*,  $\text{FINT}(k,n,\text{NY},j,l)$ ). 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

$$h_\gamma = \frac{h_L F_{3n} + h_e F_{4n}}{F_{1n} + h_L h_e F_{2n}} \quad (5.120)$$

$$h_{e'} = \frac{h_e(F_{1n} + F_{5n}) + h_L F_{2n}}{F_{1n} + h_L h_e F_{2n}} \quad (5.121)$$

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

$$W = \frac{\alpha m^2 \xi^2}{2E^2} \sum_{n=1}^{\infty} \int_0^{\omega_n} d\omega \int_0^{2\pi} \frac{d\phi}{2\pi} [f_{0n} + f_{1n} \bar{\xi}_1 + f_{3n} \bar{\xi}_3]. \quad (5.122)$$

- $\bar{\xi}_3, \bar{\xi}_1$  The Stokes parameter of the final photon (based on the polarization plane of the initial laser).
- $\phi$  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

$$f_{0n} = -\frac{|A_n^{(0)}|^2}{\xi^2} + \left(1 - x + \frac{1}{1-x}\right) \left[|A_n^{(1)}|^2 - A_n^{(0)}A_n^{(2)}\right], \quad (5.123)$$

$$f_{1n} = \frac{|A_n^{(0)}|^2}{\xi^2} \Theta^2 \sin 2\phi + 2\sqrt{2} \frac{\Theta}{\xi} \sin \phi A_n^{(0)}A_n^{(1)}, \quad (5.124)$$

$$f_{3n} = -(1 + 2\Theta^2 \sin^2 \phi) \frac{|A_n^{(0)}|^2}{\xi^2} + 2 \left[|A_n^{(1)}|^2 - A_n^{(0)}A_n^{(2)}\right]. \quad (5.125)$$

where  $A_n^{(s)}$  is defined by

$$A_n^{(s)} = \oint \frac{d\phi}{2\pi} \cos^s \phi e^{i[n\phi - \alpha_1 \sin \phi + (\alpha_2/2) \sin(2\phi)]}, \quad (s = 0, 1, 2) \quad (5.126)$$

with arguments

$$\alpha_1 = e \left( \frac{a \cdot p}{k \cdot p} - \frac{a \cdot p'}{k \cdot p'} \right) = -2\sqrt{2} \frac{\xi}{\Lambda} \frac{x\Theta}{1-x} \cos \phi,$$

$$\alpha_2 = \frac{e^2 a^2}{4} \left( \frac{1}{k \cdot p} - \frac{1}{k \cdot p'} \right) = \frac{\xi^2}{\Lambda} \frac{x}{1-x}.$$

( $a^\mu$  is the vector potential  $A^\mu$  with the phase factor  $e^{-ik_L x}$  omitted.)

The algorithm is more complicated than the circular polarization case because of one more variable  $\phi$  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.
- $\omega, \omega_L, \epsilon, \epsilon'$  Energies of the initial photon, laser photon, final electron and positron.
- $\xi$  Laser intensity parameter
- $\eta$  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  $n$ :  $u_n = n\eta / (1 + \xi^2)$ .
- $x_n$   $x_n = \frac{1}{2} [1 - \sqrt{1 - 1/u_n}]$

⊖ Lorentz invariant variable defined by

$$\Theta \equiv \sqrt{4\eta n x(1-x) - (1+\xi^2)} = \sqrt{(1+\xi^2) \left( \frac{u_n}{u} - 1 \right)} \quad (5.127)$$

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

$$p = xk + \left[ n(1-x) - \frac{4\xi^2}{\eta x} \right] k_L + m\Theta(e_1 \cos \phi + e_2 \sin \phi), \quad (5.128)$$

$$p' = (1-x)k + \left[ nx - \frac{4\xi^2}{\eta(1-x)} \right] k_L - m\Theta(e_1 \cos \phi + e_2 \sin \phi). \quad (5.129)$$

Here,  $\phi$  is the azimuthal scattering angle in a head-on frame.  $e_1^\mu$  is a spatial vector perpendicular to  $\mathbf{k}_L$  (and  $k$ ) and  $e_2^\mu$  is defined by

$$e_2^\mu \equiv \frac{\epsilon_{\mu\nu\alpha\beta} e_2^\nu k^\alpha k_L^\beta}{k \cdot k_L} = \left[ \begin{array}{c} 0 \\ \mathbf{e}_1 \times (\mathbf{k}\omega_L - \omega\mathbf{k}_L) / k \cdot k_L \end{array} \right]. \quad (5.130)$$

These vectors satisfy

$$k \cdot e_j = k_L \cdot e_j = 0, \quad e_j \cdot e_j = -1, \quad (j = 1, 2), \quad e_1 \cdot e_2 = 0. \quad (5.131)$$

#### 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

$$W = \frac{\alpha m^2 \xi^2}{2\omega} \sum_{n=1, n > (1+\xi^2)/\eta}^{\infty} \int_{x_n}^{1-x_n} dx [G_{1n} + h_L h_\gamma G_{3n} + \bar{h}_e (h_L G_{2n} + h_\gamma G_{4n})] \quad (5.132)$$

$h_\gamma$  Initial photon helicity

$h_L$  Laser helicity

$h_e$  Final electron helicity

$\bar{h}_e$  ‘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

$$W = \frac{\alpha m^2 \xi^2}{\omega} \sum_{n=1, n > (1+\xi^2)/\eta}^{\infty} \int_{x_n}^{1-x_n} dx [G_{1n} + h_L h_\gamma G_{3n}] \quad (5.133)$$

The functions  $G_{kn}$ 's are defined by

$$G_{1n} = \frac{1}{\xi^2} J_n^2 + \frac{1}{2} (2u - 1) (J_{n-1}^2 + J_{n+1}^2 - 2J_n^2) \quad (5.134)$$

$$G_{2n} = (1 - 2x) 2u \left( \frac{1}{2} - \frac{u}{u_n} \right) (J_{n-1}^2 - J_{n+1}^2) \quad (5.135)$$

$$G_{3n} = -(2u - 1) \left( \frac{1}{2} - \frac{u}{u_n} \right) (J_{n-1}^2 - J_{n+1}^2) \quad (5.136)$$

$$G_{4n} = \frac{1}{x \xi^2} J_n^2 - u(1 - 2x) (J_{n-1}^2 + J_{n+1}^2 - 2J_n^2) \quad (5.137)$$

These formulas can be obtained from those of  $F_{kn}$  for the Compton process by the replacement

$$\begin{aligned} \omega &\rightarrow -\omega, & \mathcal{E} &\rightarrow -\epsilon \\ h_\gamma &\rightarrow -h_\gamma, & h_e &\rightarrow -h_e. \end{aligned} \quad (5.138)$$

This implies

$$\begin{aligned} v &\rightarrow -1/(1 - x), & v_n &\rightarrow -4xu_n \\ v/v_n &\rightarrow u/u_n, & z_n &\rightarrow z_n. \end{aligned} \quad (5.139)$$

For convenience, we have changed the sign as

$$F_{1n} \rightarrow -G_{1n}, \quad F_{2n} \rightarrow +G_{2n}, \quad F_{3n} \rightarrow +G_{3n}, \quad F_{4n} \rightarrow -G_{4n}. \quad (5.140)$$

### 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  $\xi^2$ ) are used in **CAIN** as the independent variables instead of  $(\xi, \eta, x)$ :

- 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  $0 \leq y \leq 1$  represents electrons with energy  $\leq \omega/2$  and  $-1 \leq y \leq 0$  those  $\geq \omega/2$ . Since  $G_{1n}$  and  $G_{3n}$  are even functions of  $y$ , only the part  $y \geq 0$  is tabulated.

The mesh for  $q$  cannot be equally spaced. Select  $q$ 's such that all the threshold points  $1/n$  ( $n = n_{min}, n_{min} + 1, \dots, NPH, n_{min} = \text{integer part of } 1/ETAMAX$ ) are included, that  $q$  are equally spaced between successive thresholds, and that the spaces are not very different. Thus, the total number of  $q$ 's may not exactly equal to  $NQ$ .

The functions stored in the array  $\mathbf{GG}(k,n,i,j,l)$  are  $G_{1n}$  ( $k=1$ ) and  $G_{3n}$  ( $k=2$ ). The integral  $\mathbf{GINT}(k,n,i,j,l)$  is calculated as in the Compton case. The sum of  $\mathbf{GINT}(k,n,\mathbf{NY},j,l)$  over  $n$  does not make sense because the result would be a quite discontinuous function of  $\xi$  and  $q$ , which makes the interpolation inaccurate. However, for given  $q$ , the sum from  $n = n(q)$  to  $\mathbf{NPH}$  is continuous where  $n(q)$  is the minimum integer which does not exceed  $1/q$ . Thus, the sum over  $n = n$  to  $\mathbf{NPH}$  is stored in  $\mathbf{GALL}(k,n,j,l)$ .

For given initial condition, calculate  $\xi$  and  $q$ . Then, interpolate  $\mathbf{GALL}(k,n(q),j,l)$  for  $j$  and  $l$  and calculate the total probability  $P$  by summing for appropriate polarization:

$$P = C_0 \times \mathbf{GALL}(*), \quad \mathbf{GALL}(*) = \mathbf{GALL}_{k=1} + \mathbf{GALL}_{k=2}, \quad C_0 = \frac{\alpha m^2 \xi^2 \Delta t}{\omega}. \quad (5.141)$$

If this is larger than the value defined by  $\mathbf{PMAX}$  operand of  $\mathbf{LASERQED}$  command, the time interval is subdivided such that  $P$  becomes less than  $\mathbf{PMAX}$ .

Generate a uniform random number  $r_1$  and reject an event if  $r_1 \geq P$ . In this case the helicity of the photon should change to

$$h_{\gamma, \text{new}} = \frac{h_\gamma(1 - P_1) - h_L P_2}{1 - P}, \quad P_k = C_0 \times \mathbf{GALL}_k, \quad (k = 1, 2). \quad (5.142)$$

If  $r_1 < P$ , decide to create a pair. The number of laser photons to absorb is determined from  $\mathbf{GINT}(*,n,\mathbf{NY},j,l)$  as in the Compton case.

To determine the electron energy, generate another random number  $r_2$  in the range  $(-1,+1)$ . Then,  $y$  ( $0 \leq y \leq 1$ ) is determined from  $|r_2|$  as in the Compton case. Adopt  $-y$  instead of  $y$  if  $r_2 \leq 0$ .

The helicity of electron is given by

$$h_e = \frac{h_L G_{2n} + h_\gamma G_{4n}}{G_{1n} + h_L h_\gamma G_{3n}}. \quad (5.143)$$

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  $\rho$ , the critical energy of synchrotron radiation is given by

$$E_c = \bar{\Upsilon} E_0, \quad \bar{\Upsilon} \equiv \frac{3}{2} \Upsilon, \quad \Upsilon = \gamma^2 \frac{\lambda_C}{\rho} = \gamma \frac{B}{B_{Sch}} = \gamma \frac{\mathcal{E}}{\mathcal{E}_{Sch}} \quad (5.144)$$

where  $\lambda_C$  is the Compton wavelength,  $B_{Sch} = m^2/e = 4.4 \times 10^9 \text{Tesla}$  is Schwinger's critical field and  $\mathcal{E}_{Sch} = cB_{Sch} = 1.32 \times 10^{18} \text{V/m}$ .

The energy spectrum of emitted photons is given by the Sokolov-Ternov formula. (Radiation angle is not included in  $\mathbf{CAIN}$ . 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

$$dW = \frac{\alpha m}{\sqrt{3}\pi\gamma} F_{00} dx, \quad F_{00} = Ki_{5/3}(z) + \frac{x^2}{1-x} K_{2/3}(z). \quad (5.145)$$

Here,  $\alpha$  is the fine structure constant and

$$z = \frac{E_\gamma}{E_c} \frac{1}{1 - E_\gamma/E_0} = \frac{E_\gamma E_0}{E_c E'} = \frac{1}{\Upsilon} \frac{E_\gamma}{E'} = \frac{1}{\Upsilon} \frac{x}{1-x}. \quad (5.146)$$

$E' = E_0 - E_\gamma = E_0(1-x)$  is the final electron energy,  $K_\nu$  the modified Bessel function and  $Ki_\nu$  is its integral:

$$Ki_\nu(z) = \int_z^\infty K_\nu(z) dz. \quad (5.147)$$

The function  $F_{00}$  is available as a **CAIN** function `FuncBS`, and so are some of the  $K_\nu$ 's and  $Ki_\nu$ 's. (See Sec.2.6).

## 5.9.2 Algorithm of event generation

The random number generation using the acceptance-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  $E_\gamma \rightarrow 0$ , 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.<sup>6</sup>

$$x = \frac{\bar{\Upsilon} y^3}{1 - y^3 + \frac{1}{2} \bar{\Upsilon} (1 + y^6)}, \quad (0 < y < 1), \quad (5.148)$$

The number of photons during a time interval  $\Delta t$  in the photon energy range  $(y, y+dy)$  is then given by

$$dn_\gamma = p_0 G(\Upsilon, y) dy, \quad (5.149)$$

where

$$p_0 = \frac{c_0}{\sqrt{3}\pi} \frac{1}{(1 + \frac{1}{2} \bar{\Upsilon})^{1/3}} \frac{\alpha \gamma \Delta t}{\rho}, \quad \left( \frac{\gamma \Delta t}{\rho} = \frac{\Delta t}{\lambda_C} \frac{\mathcal{E}}{\mathcal{E}_{Sch}} \right) \quad (5.150)$$

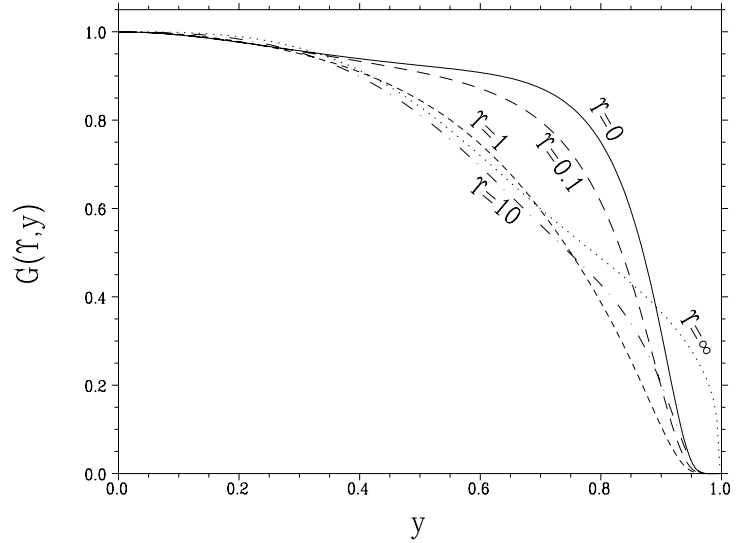
$$G(\Upsilon, y) = \frac{(1 + \frac{1}{2} \bar{\Upsilon})^{1/3}}{c_0 \bar{\Upsilon}} \frac{dx}{dy} F_{00}, \quad c_0 = \frac{9}{2^{1/3}} \Gamma(2/3). \quad (5.151)$$

The function  $G(\Upsilon, y)$  is less than or equal to unity for any  $\Upsilon$  and  $y$ . It is plotted in Fig.5.8.

The photon generation in **CAIN** proceeds in the following way.

<sup>6</sup> The definition of  $y$  has changed since **ABEL** and **CAIN2.43** in order to keep high efficiency even when  $\Upsilon$  is extremely large.

Figure 5.8: Function  $G(\Upsilon, y)$  for various values of  $\Upsilon$ . Unpolarized case only.



- (1) Calculate  $p_0$ <sup>7</sup> for given field, electron energy, and time interval  $\Delta t$ .
- (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.(5.148).

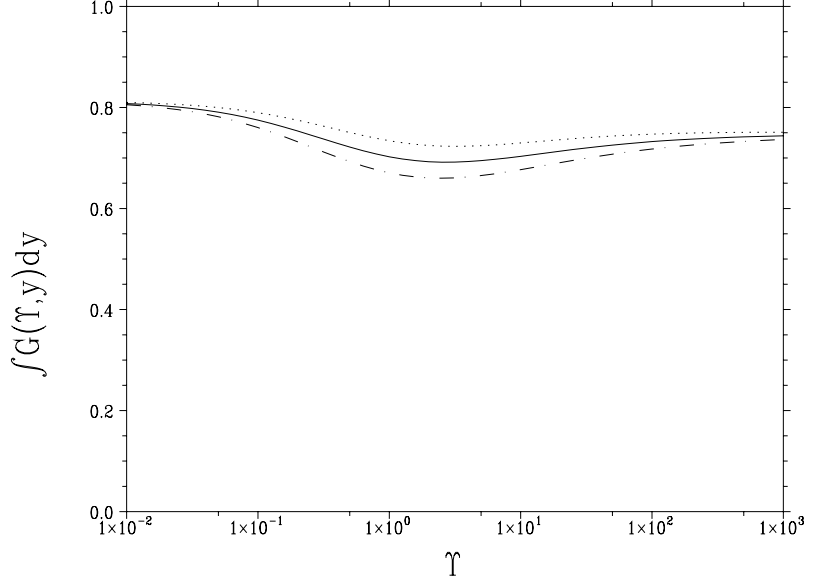
The cases when accepted in (3) but rejected in (6) cause a waste of time because the calculation of  $G(\Upsilon, y)$  is the most time consuming. The probability to be accepted in (6) is plotted in Fig.5.9 is given by  $\int_0^1 G(\Upsilon, y) dy$  and is plotted as a function of  $\Upsilon$ . One finds the probability is very high for any  $\Upsilon$  owing to the choice of the variable  $y$ .

### 5.9.3 Polarization

Polarization effects have been added since **CAIN2.1**. In order to describe polarizations, let us introduce an orthonormal basis vector  $(\mathbf{e}_1, \mathbf{e}_2, \mathbf{e}_v)$ . Here,  $\mathbf{e}_v$  is the unit vector along the initial electron velocity,  $\mathbf{e}_1$  the unit vector along the direction of the transverse component of acceleration, and  $\mathbf{e}_2 = \mathbf{e}_v \times \mathbf{e}_1$ . If the acceleration is due to a transverse

<sup>7</sup> This must be much smaller than 1. Otherwise, the probability of emitting more than one photon during  $\Delta t$  cannot be ignored. The maximum  $p_0$  is defined by the keyword **PMAX**. When  $\Upsilon$  is very large,  $p_0$  is suppressed by the factor  $(1 + \frac{1}{2}\Upsilon)^{1/3}$ . However, in defining  $\Delta t$  (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 acceptance probability in the step (6) as a function of  $\Upsilon$ . The solid line is the unpolarized case. The dot-dash and dotted lines are polarized cases with  $\zeta_i \cdot \mathbf{e}_2 = 1$  and  $-1$ , respectively.



magnetic field,  $-\mathbf{e}_2$  is the unit vector along the magnetic field (times the sign of charge). The transition rate from the initial electron polarization  $\zeta_i$  to the final polarization  $\zeta_f$  with the photon Stokes parameter  $\xi$  (based on the basis vector  $(\mathbf{e}_1, \mathbf{e}_2, \mathbf{e}_v)$ ) is

$$\begin{aligned}
 dW = \frac{\alpha dE_\gamma}{4\sqrt{3}\pi\gamma^2} \times & \left\{ F_{00}(1 + \zeta_i \cdot \bar{\zeta}_f) - xK_{1/3}(\mathbf{e}_2 \cdot \zeta_i) - \frac{x}{1-x}K_{1/3}(\mathbf{e}_2 \cdot \bar{\zeta}_f) \right. \\
 & - \frac{x^2}{1-x} \left[ (\mathbf{e}_v \cdot \zeta_i)(\mathbf{e}_v \cdot \bar{\zeta}_f)Ki_{1/3} + (\zeta_i \cdot \bar{\zeta}_f - \mathbf{e}_v \cdot \zeta_i \mathbf{e}_v \cdot \bar{\zeta}_f)K_{2/3} \right] \\
 & \left. + \frac{x}{1-x}K_{1/3}(\mathbf{e}_1 \cdot \zeta_i)\bar{\xi}_1 - \frac{x(2-x)}{2(1-x)}K_{2/3}(\mathbf{e}_v \cdot \zeta_i)\bar{\xi}_2 + \left[ K_{2/3} - \frac{x}{1-x}K_{1/3}(\mathbf{e}_2 \cdot \zeta_i) \right] \bar{\xi}_3 \right\}
 \end{aligned} \quad (5.152)$$

where  $F_{00}$  is defined in eq.(5.145) and the argument of the Bessel functions is  $z$ . We omitted the terms involving  $\bar{\zeta}_f$  and  $\bar{\xi}$  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  $\zeta_f$  and  $\xi$ .)

The radiation energy spectrum summed over the final polarization is given by eq.(5.145) with  $F_{00}$  replaced by

$$F_0 = F_{00} - xK_{1/3}(\mathbf{e}_2 \cdot \zeta_i). \quad (5.153)$$

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  $\mathbf{e}_2 \cdot \zeta_i = -1$  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,

$$\zeta_f = \frac{1}{F_0} \left\{ F_{00}\zeta_i - \frac{x}{1-x}K_{1/3}\mathbf{e}_2 - \frac{x^2}{1-x} \left[ (\mathbf{e}_v \cdot \zeta_i)\mathbf{e}_v Ki_{1/3} + [\zeta_i - (\mathbf{e}_v \cdot \zeta_i)\mathbf{e}_v]K_{2/3} \right] \right\} \quad (5.154)$$

$$\xi_1 = \frac{x}{1-x} \frac{K_{1/3}}{F_0} (\mathbf{e}_1 \cdot \boldsymbol{\zeta}_i), \quad \xi_2 = -\frac{x(2-x)}{2(1-x)} \frac{K_{2/3}}{F_0} (\mathbf{e}_v \cdot \boldsymbol{\zeta}_i), \quad \xi_3 = \frac{K_{2/3} - \frac{x}{1-x} K_{1/3} (\mathbf{e}_2 \cdot \boldsymbol{\zeta}_i)}{F_0} \quad (5.155)$$

In the case when the event generation is rejected, the polarization of the electron must be changed according to eq.(5.12):

$$\boldsymbol{\zeta}_f = \boldsymbol{\zeta}_i + [\mathbf{e}_2 - (\mathbf{e}_2 \cdot \boldsymbol{\zeta}_i) \boldsymbol{\zeta}_i] \int_0^1 \frac{\alpha m}{\sqrt{3}\pi\gamma} x K_{1/3}(z) dx. \quad (5.156)$$

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  $\bar{\boldsymbol{\zeta}}_f$  (see eq.(5.14)):

$$\frac{x}{1-x} K_{1/3} - x K_{1/3} = \frac{x^2}{1-x} K_{1/3}. \quad (5.157)$$

When  $\Upsilon$  is small ( $\lesssim 10^{-5}$  in storage rings), each term on the left hand side is proportional to  $\Upsilon$  whereas the right hand side is  $\Upsilon^2$  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 account 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

$$\chi = \frac{E_\gamma}{mc^2} \frac{B}{B_{Sch}} = \frac{E_\gamma}{mc^2} \frac{\mathcal{E}}{\mathcal{E}_{Sch}}, \quad (5.158)$$

where  $E_\gamma$  is the energy of the initial photon. ( $B_{Sch}$  and  $\mathcal{E}_{Sch}$  are defined in eq.(5.144).) The probability of the process is exponentially small ( $\propto e^{-8/(3\chi)}$ ) when  $\chi$  is small.

Let us denote the energy and polarization vector of initial photon and final positron/electron by  $E_\gamma$ ,  $E_\pm$  ( $E_+ + E_- = E_\gamma$ ),  $\boldsymbol{\xi}$ , and  $\boldsymbol{\zeta}_\pm$ . The transition rate is obtained by the following replacement in the formula (5.152):

$$\begin{aligned} E_\gamma &\rightarrow -E_\gamma, & E_0 &\rightarrow -E_+, & E' &\rightarrow E_-, & E_\gamma^2 dE_\gamma &\rightarrow -E_+^2 dE_+, \\ (\bar{\xi}_1, \bar{\xi}_2, \bar{\xi}_3) &\rightarrow (\xi_1, -\xi_2, \xi_3), \\ z = \frac{2}{3\Upsilon} \frac{E_\gamma}{E'} &\rightarrow \eta = \frac{2}{3\chi} \frac{E_\gamma^2}{E_+ E_-}. \end{aligned} \quad (5.159)$$

Ignoring the terms related to the polarization correlation between the final electron and positron, we obtain

$$\begin{aligned} dW = \frac{\alpha m^2 dE_+}{4\sqrt{3}\pi E_\gamma^2} \times &\left\{ F_{CP} - K_{2/3}\xi_3 + \frac{E_\gamma(E_+ - E_-)}{2E_+ E_-} K_{2/3} \mathbf{e}_n \cdot (\bar{\boldsymbol{\zeta}}_+ - \bar{\boldsymbol{\zeta}}_-) \xi_2 \right. \\ &+ K_{1/3} \mathbf{e}_2 \cdot \left( \frac{E_\gamma}{E_+} \bar{\boldsymbol{\zeta}}_+ + \frac{E_\gamma}{E_-} \bar{\boldsymbol{\zeta}}_- \right) + K_{1/3} (\mathbf{e}_1 \xi_1 + \mathbf{e}_2 \xi_3) \cdot \left( \frac{E_\gamma}{E_-} \bar{\boldsymbol{\zeta}}_+ + \frac{E_\gamma}{E_+} \bar{\boldsymbol{\zeta}}_- \right) \left. \right\} \end{aligned} \quad (5.160)$$

where  $F_{CP}$  is defined as

$$F_{CP} = Ki_{1/3} + \left( \frac{E_-}{E_+} + \frac{E_+}{E_-} \right) K_{2/3}. \quad (5.161)$$

$K_\nu$  is the modified Bessel function and  $Ki_\nu$  is defined in eq.(5.147). Their arguments are  $\eta$  defined in eq.(5.159).

The transition rate summed over the final polarization is

$$dW = \frac{\alpha m^2 dE_+}{\sqrt{3}\pi E_\gamma^2} (F_{CP} - K_{2/3}\xi_3). \quad (5.162)$$

The function  $F_{CP} - K_{2/3}\xi_3$  is plotted in Fig.5.10 as a function of  $E_+/E_\gamma$ . 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).

## 5.10.2 Algorithm of event generation

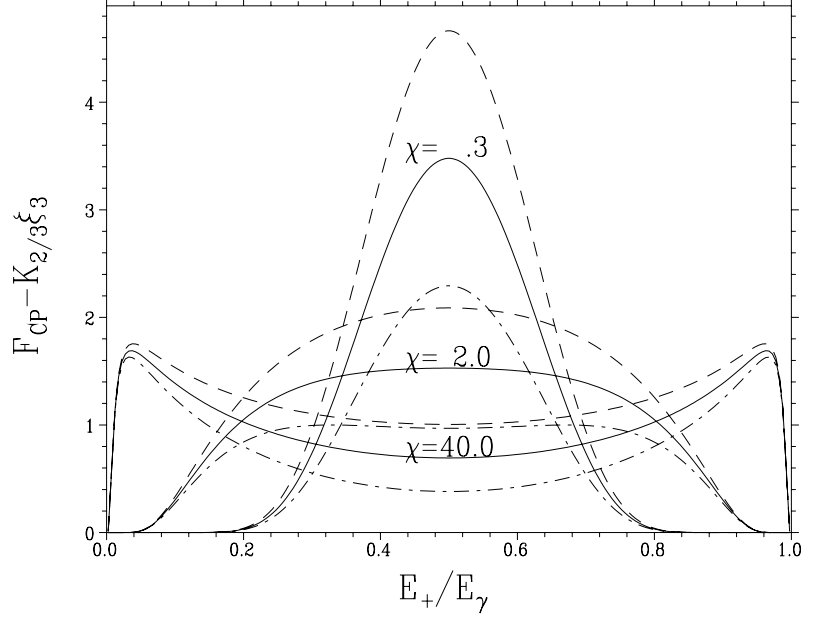
The total rate for given  $\boldsymbol{\xi}$  is approximately

$$W \approx W_{app} \equiv \frac{\alpha m^2}{E_\gamma} U, \quad U(\chi, \xi_3) \equiv e^{-\eta_0} \left\{ \frac{\chi}{[c_1^3 + c_2^3 \chi]^{1/3}} - \frac{\xi_3 \chi}{[(3c_1)^3 + (5c_2)^3 \chi]^{1/3}} \right\} \quad (5.163)$$

where

$$\eta_0 \equiv \frac{8}{3\chi}, \quad c_1 = \frac{16\sqrt{2}}{3\sqrt{3}}, \quad c_2 = \frac{7\pi}{5} \frac{2^{2/3}}{3^{2/3}} \frac{1}{[\Gamma(5/6)]^2}.$$

Figure 5.10: Function  $F_{CP} - K_{2/3}\xi_3$  for three values of  $\chi$ . 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.



In order to make the spectrum function flatter, **CAIN** introduces the variable  $y$  ( $-1 < y < 1$ ) instead of  $x = E_+/E_\gamma$  ( $0 < x < 1$ ):

$$x = \frac{1}{2} \left( 1 + \operatorname{sgn} y \sqrt{\frac{y'}{\eta_0 + y'}} \right), \quad y' = -\log \left\{ 1 - \frac{y^2}{[1 + (1 - y^2)/(2\sqrt{\eta_0})]^2} \right\}. \quad (5.164)$$

The spectrum function with respect to  $y$  then becomes

$$\frac{dW}{d(y/2)} = \frac{\alpha m^2}{E_\gamma} c_3 U(\chi, \xi_3) \tilde{F}(y, \chi, \xi_3), \quad \tilde{F} \equiv \frac{(F_{CP} - K_{2/3}\xi_3)/\sqrt{3}\pi}{c_3 U} \frac{dx}{d(y/2)}, \quad (5.165)$$

where  $y/2$  is used because the range of  $y$  is 2. The constant

$$c_3 = \frac{7\sqrt{\pi}}{6\Gamma(5/6)\Gamma(2/3)} = 1.35286 \dots$$

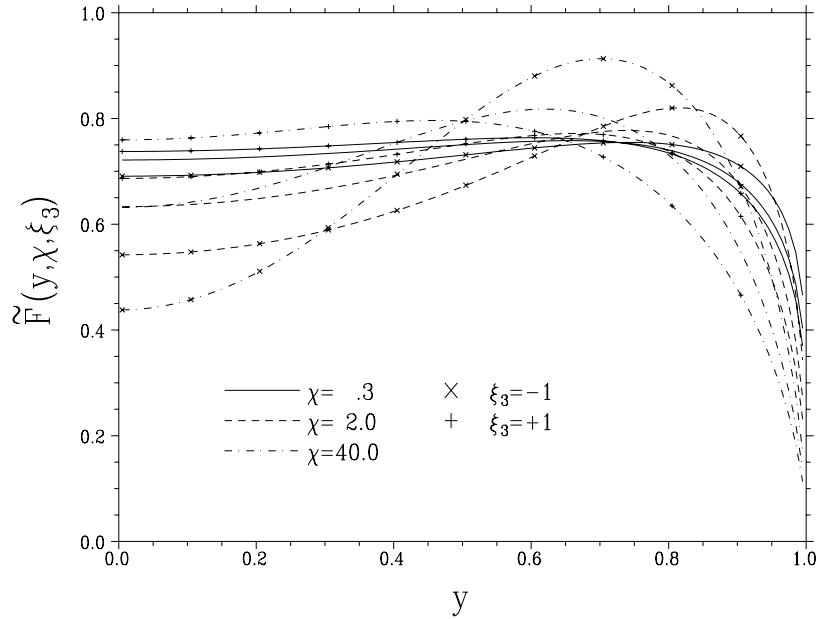
is chosen so that  $\tilde{F} \leq 1$  for any  $(y, \chi, \xi_3)$ .  $\tilde{F}(y, \chi, \xi_3)$  is plotted in Fig. 5.11.

Now, the event generation goes as

- (1) Compute  $\chi$  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.(5.164) and  $E_+ = xE_\gamma$ ,  $E_- = (1 - x)E_\gamma$ .



Figure 5.11: Function  $\tilde{F}(y, \chi, \xi_3)$  as a function of  $y$  for three values of  $\chi = 0.3, 2, 40$  and for  $\xi_3 = 0, \pm 1$ . The parameter  $\chi$  is indicated by the line mode and  $\xi_3$  is by crosses (no cross for  $\xi_3 = 0$ ).



(7) Compute the final polarization of  $e^\pm$  from

$$(F_{CP} - K_{2/3}\xi_3)\zeta_+ = \frac{1}{x}K_{1/3}\mathbf{e}_2 + \frac{1}{1-x}K_{1/3}(\xi_1\mathbf{e}_1 + \xi_3\mathbf{e}_2) + \frac{2x-1}{2x(1-x)}K_{2/3}\xi_1(\mathbf{e}_n, 166)$$

The formula for  $\zeta_-$  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):

$$\xi_{1,fin} = \xi_1 - a\xi_3\xi_1, \quad \xi_{2,fin} = \xi_2 - a\xi_3\xi_2, \quad \xi_{3,fin} = \xi_3 + a(1 - \xi_3^2), \quad (5.167)$$

$$a = \frac{\alpha m^2 \Delta t}{\sqrt{3}\pi E_\gamma} \int_0^1 K_{2/3} \left( \frac{2}{3\chi} \frac{1}{x(1-x)} \right) dx.$$

The error of the formula (5.163) 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  $\gamma$  that have to be simulated. The present version of **CAIN** include the following processes:

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

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 particles 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

$$\frac{d\sigma_{BW}}{dc} = \frac{\pi}{2} r_e^2 \frac{m^2}{\omega^2} f, \quad f = \frac{p}{\omega} (f_0 - f_2 h) \quad (5.168)$$

with

$$f_0 = \frac{\omega^2 + p^2 c^2}{\omega^2 - p^2 c^2} + \frac{1}{2} \left[ 1 - \left( 1 - \frac{2m^2}{\omega^2 - p^2 c^2} \right)^2 \right] \quad (5.169)$$

$$f_2 = \frac{\omega^2 + p^2 c^2}{\omega^2 - p^2 c^2} \left( 1 - \frac{2m^2}{\omega^2 - p^2 c^2} \right) \quad (5.170)$$

where

- $\omega, p$  Energy and momentum of final electron in the center-of-mass frame.
- $c$  Cosine of the scattering angle  $\theta$  of the final electron in the center-of-mass frame.
- $h$  Product of circular polarizations of the two initial photons.

The total crosssection is

$$\sigma_{BW} = \pi r_e^2 \frac{m^2}{\omega^2} G \quad (5.171)$$

where

$$G = \int_0^1 f dc = 2 \left( 1 - h + \frac{2a^2 - 1}{2a^4} \right) \sinh^{-1} b + \frac{b}{a} \left[ -\left(1 + \frac{1}{a^2}\right) + 3h \right], \quad (5.172)$$

where

$$a = \omega/m, \quad b = p/m, \quad a^2 = b^2 + 1.$$

Events are generated by the following algorithm using inverse function.

- (a) Compute  $\omega$ ,  $p$ ,  $a$ ,  $b$ ,  $G$  and  $\sigma$  for given initial parameters (reject if  $a \leq 1$ , *i.e.*, below threshold) and calculate the event probability for the given time step

$$P = \frac{w_1 w_2 \sigma_{BW} \Delta t}{w V}$$

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,  $\Delta t$  the time interval and  $V$  the volume in which the macro photons are located.

- (b) If  $P$  is too large (say,  $> 0.1$ ), divide the interval  $\Delta t$  (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  $r_1 \geq P$ .
- (d) Generate another random number  $r_2 \in (-1, 1)$  and solve the equation

$$2 \left( 1 - h + \frac{2a^2 - 1}{2a^4} \right) z - (1 - h) \tanh z + \frac{2a^2 h - 1}{a^4} \sinh z \cosh z = |r_2| G \quad (5.173)$$

with respect to  $z$ . Here  $z$  is defined by  $|c| = |\cos \theta| = (a/b) \tanh z$  ( $0 \leq z \leq \sinh^{-1} b$ ). 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

$$\mathbf{p}_\perp = p \sin \theta (\mathbf{e}_1 \cos \phi + \mathbf{e}_2 \sin \phi) \quad (5.174)$$

where  $\mathbf{e}_1$  and  $\mathbf{e}_2$  are arbitrary unit vectors perpendicular to  $\mathbf{e}_3$ , the unit vector along the initial photon momentum in the center-of-mass frame. The latter is given by

$$\mathbf{e}_3 = \frac{(\omega + \omega_2) \mathbf{k}_1 - (\omega + \omega_1) \mathbf{k}_2}{\omega(2\omega + \omega_1 + \omega_2)}$$

where  $(\omega_j, \mathbf{k}_j)$  are the energy momentum of the photons in the original frame. The value of  $\sin \theta$  should be computed from

$$\sin^2 \theta = \frac{b^2 - \sinh^2 z}{b^2 \cosh^2 z}, \quad (\sin \theta \geq 0)$$

rather than from  $|\cos \theta|$  because the latter is usually very close to unity when  $\omega$  is much larger than the electron rest mass.

(f) Then, the momentum of the electron in the original frame is calculated by

$$\mathbf{p} = \frac{1}{2} \left[ \left(1 + \frac{pc}{\omega}\right) \mathbf{k}_1 + \left(1 - \frac{pc}{\omega}\right) \mathbf{k}_2 \right] + \mathbf{p}_\perp + \frac{\mathbf{k} \cdot \mathbf{p}_\perp}{2\omega + \omega_1 + \omega_2} \frac{\mathbf{k}}{2\omega} \quad (5.175)$$

where  $\mathbf{k} = \mathbf{k}_1 + \mathbf{k}_2$ . 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:

$$\begin{array}{ll} \text{Bethe-Heitler} & \gamma + \text{'}\gamma\text{'} \rightarrow e^- + e^+ \\ \text{Landau-Lifshitz} & \text{'}\gamma\text{'} + \text{'}\gamma\text{'} \rightarrow e^- + e^+ \\ \text{Bremsstrahlung} & e + \text{'}\gamma\text{'} \rightarrow e + \gamma \end{array}$$

where 'γ' is a virtual photon.

Let the electron energy be  $E = m\gamma$  ( $\gamma \gg 1$ ). The number of virtual photons with energy  $\omega$  and transverse momentum  $\mathbf{q}_\perp$  is given by

$$dn = \frac{\alpha}{\pi} \frac{d\omega}{\omega} \frac{1}{\pi} \frac{\mathbf{q}_\perp^2 d\mathbf{q}_\perp}{(\mathbf{q}_\perp^2 + \omega^2/\gamma^2)^2}, \quad (|\mathbf{q}_\perp| \lesssim m) \quad (5.176)$$

where  $\alpha$  is the fine structure constant. For given  $\omega$ , 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

$$dn = \frac{\alpha}{\pi} \frac{d\omega}{\omega} K_1^2\left(\frac{\omega\rho}{\gamma}\right) \frac{\omega}{\gamma^2} \frac{d\mathbf{r}_\perp}{\pi}, \quad (\rho \gtrsim 1/m) \quad (5.177)$$

where  $\mathbf{r}_\perp$  is the transverse coordinate with respect to the parent electron,  $\rho = |\mathbf{r}_\perp|$ , and  $K_1$  the modified Bessel function.

The transverse momentum cut off  $|\mathbf{q}_\perp| \lesssim m$  (or  $\rho \gtrsim 1/m$ ) is somewhat ambiguous. 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  $\omega_{min}$  of the integration over  $\omega$  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<sup>8</sup>

$$n = \frac{\alpha}{\pi} \int_{y_{min}}^1 \frac{dy}{y} \int_y^\infty K_1^2(x) 2x dx \quad (5.178)$$

$$= \frac{\alpha}{\pi} \int_{y_{min}}^1 \frac{dy}{y} V(y), \quad (5.179)$$

with

$$V(x) = x^2[K_0(x)K_2(x) - K_1^2(x)]. \quad (5.180)$$

When  $y_{min} \ll 1$ , the total number is

$$n = \frac{\alpha}{\pi} \left[ \log^2 y_{min} - (2 \log 2 - 2\gamma_E - 1) \log y_{min} \right], \quad (5.181)$$

where  $\gamma_E = 0.577\dots$  is Euler's constant. At very high energies the number of virtual photons per electron is  $O(1)$ , in spite of the small factor  $\alpha/\pi$ , 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  $\eta$  instead of  $y$ :

$$y = \exp \left[ -\frac{\eta}{\sqrt{c + \eta}} \right], \quad \eta = \frac{1}{2} \left( \log^2 y + \sqrt{\log^4 y + 4c \log^2 y} \right). \quad (5.182)$$

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  $\eta$  is

$$\eta_{max} = \frac{1}{2} \left( \log^2 y_{min} + \sqrt{\log^4 y_{min} + 4c \log^2 y_{min}} \right). \quad (5.183)$$

Now, the spectrum with respect to  $\eta$  is

$$dn = \frac{\alpha}{\pi} G(\eta) d\eta, \quad (5.184)$$

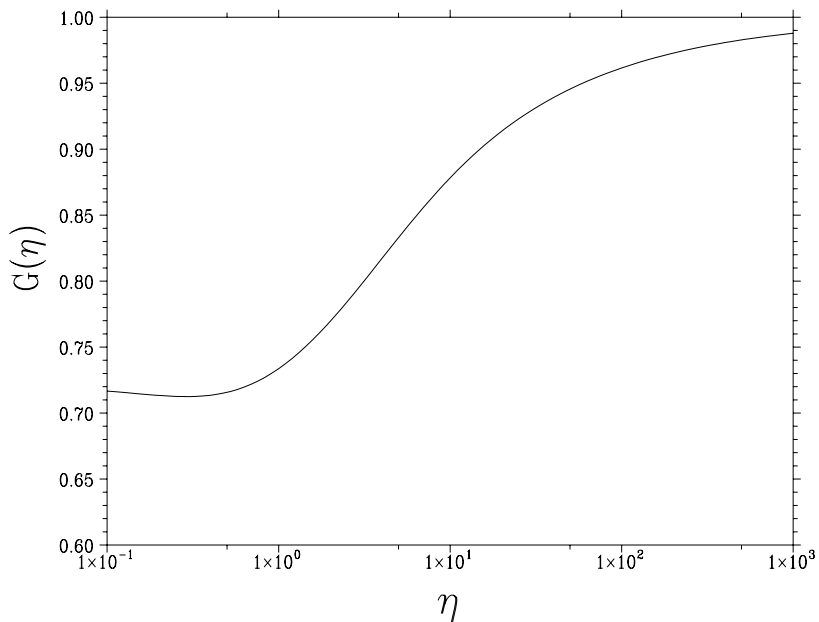
with

$$G(\eta) = \frac{2c + \eta}{2(c + \eta)^{3/2}} V(y). \quad (5.185)$$

---

<sup>8</sup> The upper limit  $y = 1$  is not regorous. The recoil effect must be taken into account when  $y$  is large.

Figure 5.12: Function  $G(\eta)$  defined in eq.(5.185). It is close to unity because only large  $\eta$  region is important.  $G(0)$  is finite and depends on the parameter  $c$ .  $G(0) < 1$  if  $c > 0.1035\dots$ . Here,  $c = 0.2$  is adopted.



For  $0 < \eta < \infty$ ,  $G(\eta) \leq 1$  and close to 1 except for the small  $\eta$  region which is unimportant in practice. Thus,

$$n = \frac{\alpha}{\pi} \int_0^{\eta_{max}} G(\eta) d\eta \leq \frac{\alpha}{\pi} \eta_{max}. \quad (5.186)$$

For given  $\eta$  (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  $y_{min}$ ,  $\eta_{max}$  and

$$P_0 \equiv \frac{\alpha \eta_{max}}{\pi w}$$

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.

- (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.(5.182) and  $\omega = Ey$ . If LOCAL option is specified, stop here and return  $\mathbf{r}_\perp = (0, 0)$ . Otherwise, calculate the value of  $V(y)$  from  $G$  using eq.(5.185).
- (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  $\rho = \lambda_e x/y$ ,  $\lambda_e$  being the Compton wave length.
- (g) Generate a random number  $r_4 \in (0, 1)$  and compute the photon coordinate  $\mathbf{r}_\perp = (\rho \cos 2\pi r_4, \rho \sin 2\pi r_4)$ .

# Appendix A

## History of Revision

There can be lots of items (in particular bug-fixes) missing here.

### A.1 CAIN2.44b

- Operand ROUND added in MAGNET and BEAMLINE commands. Enabled tracking photons added in TRANSPORT.
- Operand Ppdata introduced in LUMINOSITY command (Feb.2017)
- Operand EXIT introduced in FLAG command. (Jan.2017)
- "SUM" operand added in PLOT HISTOGRAM command. (Apr.2016)

### A.2 CAIN2.43

- The Bessel function routine BESJDJ replaced by BESJDJNEW (Dec.07.2011)
- The common block NLLSRCM replaced with module NLLSRCM. Dynamic allocation of work area for nonlinear QED introduced. Operand MWNLLSR introduced in ALLOCATE command. (Dec.08.2011)
- JUNIFORM operand added in BEAM command. (Jan.31.2012)
- In the format statement for writing beam data, "D" descriptor changed to "E" (file bmfmt.h in include directory) (May.2014)
- "GEANT4" operand added in WRITE BEAM command.

### A.3 CAIN2.42

- Added DEBUGFLAG for debugging purpose. (Jun.23.2011)
- The meaning of Pmax in LASERQED command was changed and a new keyword PSTOP was introduced. (Jun.25.2011)



## A.4 CAIN2.40

- Added DALPHADT for travelling focus. (Sep.10.2009)

## A.5 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 `vphbf1.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 directory `src/local/` was removed. The file `windows/second.c` was removed. There is no more C files. (Apr.23.2003)
- The redundant 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 MODULES. (Apr.25.2003)

## A.6 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.3.32
- 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.  
 $P0=FINSTRC/PI*HMAX/WGT \rightarrow P0=FINSTRC/PI*HMAX*WGT$   
 This bug is serious. It seems, however, 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  $\gamma$ - $\gamma$ ).
- 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.7 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 external 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 lsrgo.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)  
`CHARGO=CHARG` added just after the line  
`ELSEIF(CHARG.NE.CHARGO) THEN`
- Bugfix in cprnd2.f.  
`EXPETA=EXP(-DETA) → EXPETA=EXP(-ETA)`  
`FCP=FCOHP*DXDP/FMAX*CC0/WCOHP0 → FCP=FCOHP*DXDP/FMAX*CC0/WCOHP1`

## A.8 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.9 CAIN2.3

- The ‘expression’ greatly 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 grammar), having been replaced by the much more powerful operand SELECT for particle selection. See Sec.3.32.
- Character strings such as FILE='...' must be enclosed by apostrophes. (Old versions allowed omitting them if the string did not contain blank 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.10 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.11 CAIN2.21

- Bug fix in `donut.f` ( $\mathbf{n}$  vector)
- Random number routine changed the name: `rand`→`randcain` to avoid name conflict in some unix systems.
- `makefile` rewritten for `jlccc`. (Thanks to I. Sakai)
- `@go` rewritten for `jlccc`. (Thanks to T. Omori)

## A.12 CAIN2.2a

- Donut laser beam by axicon introduced.

## A.13 CAIN2.2

There has been a version `CAIN2.1d` 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.14 History until the version CAIN2.1b

New entries on physics

- 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.

#### New entries on user interface

- 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 because **CAIN** computes these phenomenon due to the external fields, too.<sup>1</sup> According to this change, CFQED operand was added to CLEAR command. Except for this change, input data files prepared for **CAIN2.43** can be used for **CAIN2.44b**.
- 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)

---

<sup>1</sup>Following this logic more faithfully, **CAIN** should have adopted the word SYNCHROTRONRADIATION instead of BEAMSTRAHLUNG.

- 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.  $(\xi_1, \xi_3)$  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. [85](#), [104](#), [109](#)
- [4] H. A. Tolhoek, *Rev. Mod. Phys.* **28** (1956) 277. [104](#)
- [5] Y. S. Tsai, SLAC-PUB-5924 Nov. 1992. [109](#)
- [6] T. Tauchi, K. Yokoya and P. Chen, *Part. Acc.* **41** (1993) 29.
- [7] Yabo Liu and Ping He. [47](#)

[102](#)

# Index

- \$FtoA, 18
- \$ItoA, 18
- \$NextMag, 15, 56
- \$PrevMag, 15, 56
- \$Substr, 18
- \$ToLower, 18
- \$ToUpper, 18
  
- Abs, 15
- ALLOCATE, 25
- almost real photon, 123
- Alpha, 20
- anomalous magnetic moment, 88
- ArcCos, 15
- ArcCosh, 15
- ArcSin, 15
- ArcSinh, 15
- ArcTan, 15
- ArcTanh, 15
- ARRAY, 26
- array, 16
- Atan2, 15
- AtoF, 17
- AvrEn, 18
- AvrPs, 18
- AvrPx, 18
- AvrPy, 18
- AvrS, 18
- AvrSs, 18
- AvrSx, 18
- AvrSy, 18
- AvrT, 18
- AvrX, 18
- AvrXi1, 18
- AvrXi2, 18
- AvrXi3, 18
- AvrY, 18
- axicon, 37
  
- BBFIELD, 43, 68
- BBFIELD clear, 71
- BEAM, 27
- beam field, 43, 96
- beam function, 18
- BEAMLIN, 51
- beamline, 90
- beamline coordinate, 90
- beamline functions, 20
- BeamMatrix, 18
- beamstrahlung, 42, 113
- BEAMSTRAHLUNG clear, 71
- BesJ, 21
- BesK, 21
- BesK13, 21
- BesK23, 21
- BesKi13, 21
- BesKi53, 21
- Bessel function, 21
- Beta, 20
- Bethe-Heitler process, 123
- BLOPTICS, 52
- BMT equation, 83, 88, 131
- Breit-Wheeler process, 40, 121
  
- CAIN** function, 18
- CFQED, 42
- character expression, 16
- classical electron radius, 14
- CLEAR, 70
- coherent pair creation, 42, 117
- coherent pair creation(clear), 71
- command, 11, 24
- comment, 10
- compilation of **CAIN**, 76
- Compton scattering, 40, 103
- Compton wavelength, 14
- Cos, 15



Cosh, 15  
 crab angle, 28, 86  
 crossing angle, 28  
  
 dagger †, 24  
 DBesJ, 21  
 DBesK, 21  
 Debug, 15  
 DEBUGFLAG, 73  
 differential luminosity, 44, 45  
 directory of **CAIN** file, 76, 79  
 DO, 57  
 donut, 34, 37  
 donuts, 101  
 DRIFT, 48  
  
 ECHO, 26  
 elliptic coordinate, 99  
 elliptic distribution, 28  
 ELSE, 58  
 ELSEIF, 58  
 emittance, 27, 86  
 END, 74  
 ENDDO, 57  
 endfile, 72  
 ENDIF, 58  
 ENDPUSH, 48  
 ENDTRANSPORT, 55  
 enhancement of event rate, 41, 47  
 equivalent photon approximation, 123  
 Eta, 20  
 Etaprime, 20  
 Exp, 15  
 expression, 12, 62  
 EXTERNALFIELD, 44, 49, 50  
 EXTERNALFIELD clear, 71  
  
 fast Fourier transformation, 96  
 FILE, 72  
 file close, 72  
 file open, 72  
 file rewind, 72  
 fine structure constant, 14  
 FLAG, 25  
 Frac, 15  
 FuncBS, 21, 114  
  
 FuncCP, 21, 118  
  
 Gamma, 15  
 generation, 30, 82  
  
 harmonic expansion, 98  
 HEADER, 72  
 helicity, 45  
  
 IF, 58  
 incoherent pair creation, 46, 120  
 incoherent pair creation(clear), 71  
 incoherent processes, 120  
 installation of **CAIN**, 76  
 Int, 15  
 IntFCP, 22, 118  
  
 Landau-Lifshitz, 123  
 LASER, 34, 50  
 LASER clear, 71  
 laser geometry, 99  
 laser propagation, 102  
 LaserIntensity, 21  
 LASERQED, 40  
 LASERQED clear, 71  
 LaserRange, 21  
 Log, 15  
 log scale, 65, 67  
 Log10, 15  
 longitudinal coordinate  $s$ , 81  
 longitudinal mesh, 44, 96  
 LORENTZ, 31, 49  
 Lorentz transformation, 49  
 Lum, 21  
 LumEE, 21  
 LumEEbin, 21  
 LumEEbinEdge, 21  
 LumEEH, 21  
 LumEEP, 21  
 LumH, 21  
 LUMINOSITY, 44  
 LUMINOSITY clear, 71  
 luminosity function, 21, 45  
 luminosity integration, 93  
 LumP, 21  
 LumW, 21

LumWbin, 21  
 LumWbinEdge, 21  
  
 MAGNET, 50  
 MATCHING, 53  
 Max, 15  
 meta-expression, 22, 51  
 Min, 15  
 Mod, 15  
 modified Bessel function, 21  
 MsgFile, 15, 22  
 MsgFLevel, 15  
  
 Nint, 15  
 NMacro, 18  
 nonlinear Breit-Wheeler process, 110  
 nonlinear Compton process, 106  
 NParticle, 18  
 Nu, 20  
  
 operator, 13  
 OutFile, 15, 22  
 OutFile2, 15, 22  
  
 particle selection, 74  
 Planck's constant, 14  
 PLOT, 64  
 Poisson equation, 96  
 polarization luminosity, 94  
 polarization vector, 29, 30, 81, 83  
 positional operand, 24  
 PPINT, 46  
 predefined function, 15  
 predefined parameter, 14  
 PRINT, 59  
 print array, 63  
 print parameters, 62  
 PUSH, 34, 48  
  
 Rand, 15  
 random number, 15  
 Rayleigh length, 37, 100  
 RESTORE, 72  
 rotation, 49  
 running CAIN, 78  
 running variable, 14, 64  
  
 Schwinger's critical field, 113  
 SELECT, 74  
 SET, 26  
 Sgn, 15  
 shift of origin, 49  
 SigEn, 18  
 SigPs, 18  
 SigPx, 18  
 SigPy, 18  
 SigS, 18  
 SigSs, 18  
 SigSx, 18  
 SigSy, 18  
 SigT, 18  
 SigX, 18  
 SigXi1, 18  
 SigXi2, 18  
 SigXi3, 18  
 SigY, 18  
 Sin, 15  
 Sinh, 15  
 Smesh, 15, 44, 45, 96  
 Sokolov-Ternov formula, 113  
 SPIN, 26  
 Sqrt, 15  
 standard format, 30, 60  
 statistics of beam, 60  
 Step, 15  
 Stokes parameter, 29, 30, 35, 81  
 STOP, 73  
 STORE, 72  
 Strlen, 17  
 Strstr, 18  
  
 Tan, 15  
 Tanh, 15  
 TDFile, 15, 22  
 test particle, 19, 33  
 TestEn, 19  
 TestPs, 19  
 TestPx, 19  
 TestPy, 19  
 TestS, 19  
 TestT, 19  
 TestX, 19

TestY, 19  
Thomas precession, 82  
Thomas-BMT equation, 83, 88, 131  
Time, 14, 34, 48  
time coordinate  $t$ , 81  
TMatTMat, 20  
transfer matrix, 20  
TRANSPORT, 55  
transverse mesh, 44, 93  
Twiss parameter, 20  
  
uniform distribution, 28  
user-defined parameter, 15, 26  
  
velocity of light, 14  
virtual photon, 123  
  
Weizäcker-Williams approximation, 123  
WRITE, 59  
  
zero padding, 97