IKDA 96/13-rev
hep-ph/9607454-rev
July 1996 (expanded September 1996)

Κίρκη Version 1.02β: Beam Spectra for Simulating Linear Collider Physics*

Thorsten Ohl†

Technical University of Darmstadt
Schloßgartenstr. 9
D-64289 Darmstadt
Germany

Abstract

I describe parameterizations of realistic $e^{\pm}$- and γ-beam spectra at future linear e^+e^--colliders. Emphasis is put on simplicity and reproducibility of the parameterizations, supporting reproducible physics simulations. The parameterizations are implemented in a library of distribution functions and event generators.

*Supported by Bundesministerium für Bildung, Wissenschaft, Forschung und Technologie, Germany.
†e-mail: Thorsten.Ohl@Physik.TH-Darmstadt.de

Program Summary:

- **Title of program**: Κίρκη, Version 1.02β (September 1996)
- **Program obtainable** by anonymous `ftp` from the host `crunch.ikp.physik.th-darmstadt.de` in the directory `pub/ohl/circe`.
- **Licensing provisions**: Free software under the GNU General Public License.
- **Programming language used**: Fortran77
- **Number of program lines in distributed program, including test data, etc.**: $\approx$ 1100 (excluding comments)
- **Computer/Operating System**: Any with a Fortran77 programming environment.
- **Memory required to execute with typical data**: Negligible on the scale of typical applications calling the library.
- **Typical running time**: A small fraction (typically a few percent) of the running time of applications calling the library.
- **Purpose of program**: Provide simple and reproducible, yet realistic, parameterizations of the $e^{\pm}$- and γ-beam spectra for linear colliders.
- **Nature of physical problem**: The intricate beam dynamics in the interaction region of a high luminosity linear collider at $\sqrt{s} = 500\text{GeV}$ result in non-trivial energy spectra of the scattering electrons, positrons and photons. Physics simulations require simple and reproducible, yet realistic, parameterizations of these spectra.
- **Method of solution**: Parameterization, curve fitting, Monte Carlo event generation.
- **Keywords**: Event generation, beamstrahlung, linear colliders.

ἔρχεο νῦν συφεόνδε, μετ' ἄλλων λέξο ἑταίρον.

BLOOM: *(with sinews semiflexed)* Magmagnificence!

1 Introduction

Despite the enormous quantitative success of the electro-weak standard model up to energies of 200GeV, neither the nature of electro-weak symmetry breaking (EWSB) nor the origin of mass are understood.

From theoretical considerations, we know that clues to the answer of these open questions are hidden in the energy range below $\Lambda_{\text{EWSB}} = 4\pi v \approx 3.1\text{TeV}$. Either we will discover a Higgs particle in this energy range or signatures for a strongly interacting EWSB sector will be found. Experiments at CERN's Large Hadron Collider (LHC) will shed a first light on this regime in the next decade. In the past is has been very fruitful to complement experiments at high energy hadron colliders with experiments at e^+e^--colliders. The simpler initial state allows more precise measurements with smaller theoretical errors. Lucid expositions of the physics opportunities of high energy e^+e^- colliders with references to the literature can be found in [1].

However, the power emitted by circular storage rings in form of synchrotron radiation scales like $(E/m)^4/R^2$ with the energy and mass of the particle and the radius of the ring. This cost becomes prohibitive after LEP2 and a Linear Collider (LC) has to be built instead.

Unfortunately, the "interesting" hard cross sections scale like $1/s$ with the square of the center of mass energy and a LC will have to operate at extremely high luminosities in excess of $10^{33}\text{cm}^{-2}\text{s}^{-1}$. To achieve such luminosities, the bunches of electrons and positrons have to be very dense. Under these conditions, the electrons undergo acceleration from strong electromagnetic forces from the positron bunch (and vice versa). The resulting synchrotron radiation is called *beamstrahlung* [2] and has a strong effect on the energy spectrum $D(x_1, x_2)$ of the colliding particles. This changes the observable e^+e^- cross sections

$$\frac{d\sigma_0^{e^+e^-}}{d\Omega}(s) \to \frac{d\sigma^{e^+e^-}}{d\Omega}(s) = \int_0^1 dx_1\, dx_2\, D_{e^+e^-}(x_1, x_2; \sqrt{s}) J(\Omega', \Omega) \frac{d\sigma_0^{e^+e^-}}{d\Omega'}(x_1 x_2 s) \tag{1a}$$

and produces luminosity for $e^{\pm}\gamma$ and $\gamma\gamma$ collisions:

$$\frac{d\sigma^{e^{\pm}\gamma}}{d\Omega}(s) = \int_0^1 dx_1\, dx_2\, D_{e^{\pm}\gamma}(x_1, x_2; \sqrt{s}) J(\Omega', \Omega) \frac{d\sigma_0^{e^{\pm}\gamma}}{d\Omega'}(x_1 x_2 s) \tag{1b}$$

$$\frac{d\sigma^{\gamma\gamma}}{d\Omega}(s) = \int_0^1 dx_1\, dx_2\, D_{\gamma\gamma}(x_1, x_2; \sqrt{s}) J(\Omega', \Omega) \frac{d\sigma_0^{\gamma\gamma}}{d\Omega'}(x_1 x_2 s) \tag{1c}$$

Therefore, simulations of the physics expected at a LC need to know the spectra of the $e^{\pm}$ and γ beams precisely.

Microscopic simulations of the beam dynamics are available (e.g. `ABEL`[3], `CAIN`[4] and `Guinea-Pig`[5]) and their predictions are compatible with each other. But they require too much computer time and memory for direct use in physics programs. Kiρκη provides a fast and simple parameterization of the results from these simulations. Furthermore, even if the computational cost of the simulations would be negligible, the input parameters for microscopic simulations are not convenient for particle physics applications. Due to the highly non-linear beam dynamics, the optimization of LC designs is a subtle art [6], that is best practiced by the experts. Furthermore, particle physics applications need benchmarking and easily reproducible parameterizations are required for this purpose.

The parameterizations in Kiρκη are not based on approximate solutions (cf. [7]) of the beamstrahlung dynamics. Instead, they provide a "phenomenological" description of the results from full simulations. The parameterizations are as simple as possible while remaining consistent with basic physical principles:

1. *positivity:* the distribution functions $D(x_1, x_2)$ *must not* be negative in the physical region $[0,1] \times [0,1]$.

2. *integrability:* the definite integral of the distribution functions over the physical region $[0,1] \times [0,1]$ *must* exist, even though the distributions can have singularities.

This paper is organized as follows: I start in section 2 with a discussion of the input for the microscopic simulations. In section 3 I describe the usage of the Kiρκη library and in section 4 I discuss some technical details of the implementation. After discussing the parameterizations available in version 1.02β in section 5, I conclude in section 6.

	SBAND	TESLA	XBAND	SBAND	TESLA	XBAND
E/GeV	250	250	250	500	500	500
$N_{particles}/10^{10}$	1.1	3.63	0.65	2.9	1.8	0.95
$\epsilon_x/10^{-6}$mrad	5	14	5	10	14	5
$\epsilon_y/10^{-6}$mrad	0.25	0.25	0.08	0.1	0.06	0.1
β_x^*/mm	10.98	24.95	8.00	32	25	10.00
β_y^*/mm	0.45	0.70	0.13	0.8	0.7	0.12
σ_x/nm	335	845	286	571.87	598.08	226
σ_y/nm	15.1	18.9	4.52	9.04	6.55	3.57
σ_z/μm	300	700	100	500	500	125
f_{rep}	50	5	180	50	5	180
n_{bunch}	333	1135	90	125	2270	90

Table 1: Accelerator parameters for three typical designs at $\sqrt{s} = 500$GeV and $\sqrt{s} = 1$TeV. The resulting distributions are shown in figure 1. The design efforts are currently concentrated on a 350GeV-800GeV LC. Therefore the Tesla parameters for 1TeV are slightly out of date.

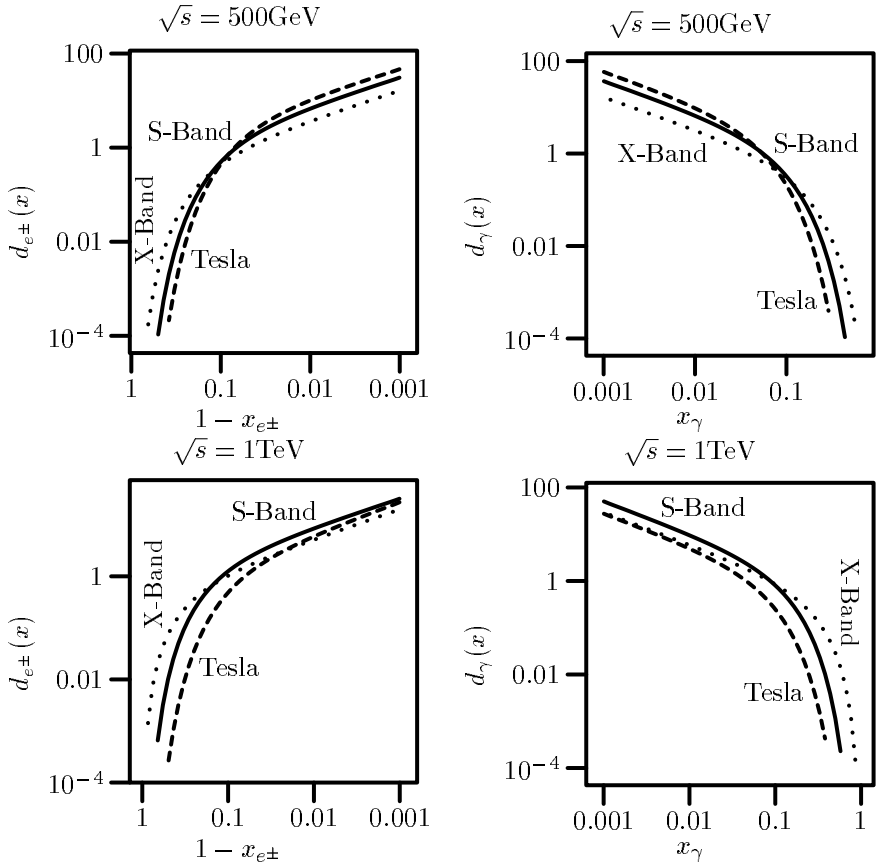

Figure 1: Version 1, revision 1996 09 02 of the factorized $e^\pm$- and γ-distributions at $\sqrt{s} = 500$GeV and $\sqrt{s} = 1$TeV in a doubly logarithmic plot. The accelerator parameters are taken from table 1.

	TESLA	TESLA	TESLA
E/GeV	175	250	400
$N_{particles}/10^{10}$	3.63	3.63	3.63
$\epsilon_x/10^{-6}$mrad	14	14	14
$\epsilon_y/10^{-6}$mrad	0.25	0.25	0.1
β^*_x/mm	25.00	24.95	15.00
β^*_y/mm	0.70	0.70	0.70
σ_x/nm	1010.94	845	668.67
σ_y/nm	22.6	18.9	9.46
σ_z/μm	700	700	700
f_{rep}	5	5	5
n_{bunch}	1135	1135	1135

Table 2: Accelerator parameters for the Tesla design at three planned [8] energies. The resulting distributions are shown in figure 2.

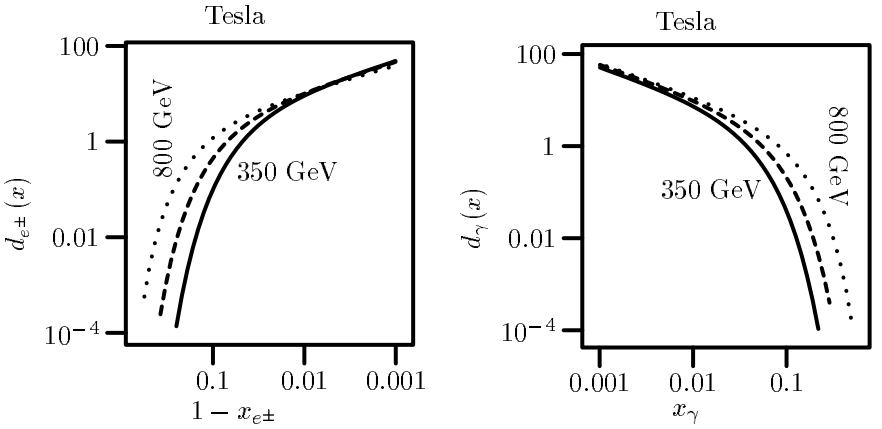

Figure 2: Version 1, revision 1996 09 02 of the factorized $e^\pm$- and γ-distributions for Tesla in a doubly logarithmic plot. The accelerator parameters are taken from table 2.

	High-$\mathcal{L}$	Low-$\mathcal{L}$	Low-ϵ_y
E/GeV	400	400	400
$N_{particles}/10^{10}$	3.63	3.63	1.800
$\epsilon_x/10^{-6}$mrad	14	14	12
$\epsilon_y/10^{-6}$mrad	0.1	0.25	0.025
β_x^*/mm	15.00	25.00	25.00
β_y^*/mm	0.70	0.70	0.50
σ_x/nm	668.67	700.00	
σ_y/nm	9.46		
σ_z/μm	700	700	500
f_{rep}	5	5	3
n_{bunch}	1135	1135	2260

Table 3: Variant accelerator parameters for the Tesla design at 800 Gev.

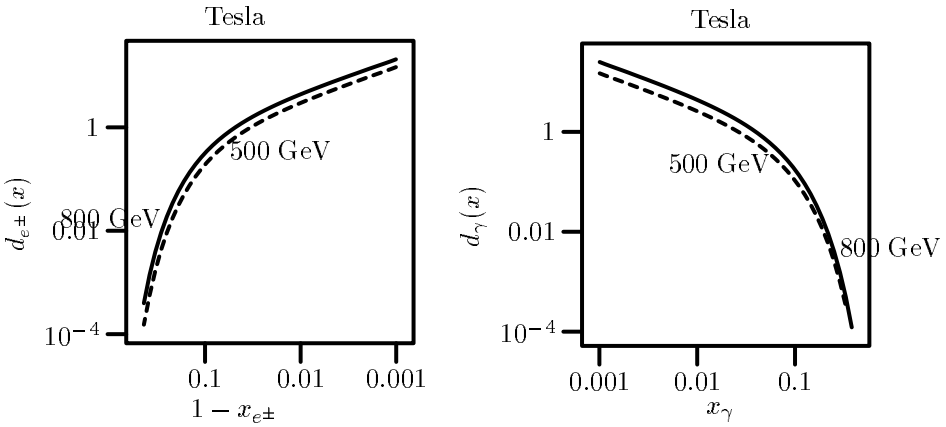

Figure 3: Version 5, revision 1998 05 05 of the factorized $e^\pm$- and γ-distributions for a high luminosity Tesla in a doubly logarithmic plot. The accelerator parameters are taken from table 4.

	TESLA	TESLA
E/GeV	250	400
$N_{particles}/10^{10}$	2	1.40
$\epsilon_x/10^{-6}$m rad	10	8
$\epsilon_y/10^{-6}$m rad	0.03	0.01
β_x^*/mm	15.00	15.00
β_y^*/mm	0.40	0.30
σ_x/nm	553	391
σ_y/nm	5	2
σ_z/μm	400	300
f_{rep}	5	3
n_{bunch}	2820	4500

Table 4: Accelerator parameters for a high luminosity Tesla design at two planned [8] energies. The resulting distributions are shown in figure 3.

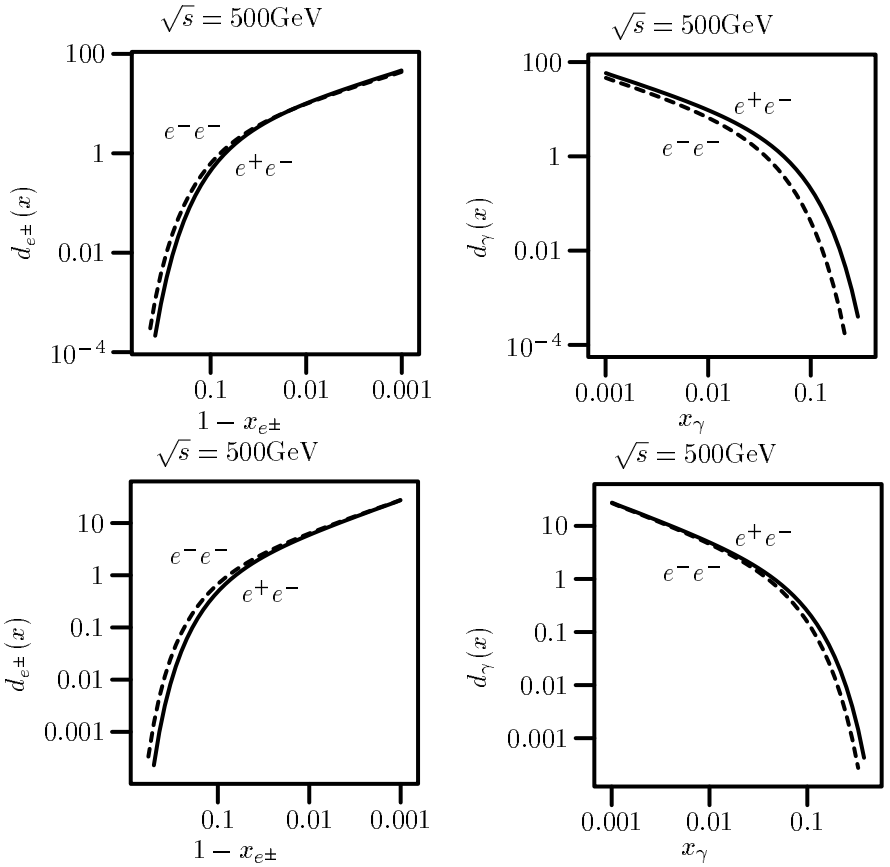

Figure 4: *Experimental:* Version 1, revision 0 of the factorized e^-- and γ-distributions for Tesla-e^-e^- in a doubly logarithmic plot. The accelerator parameters are taken from table 2 and have *not* been endorsed for use in an e^-e^--machine yet!.

2 Parameters

The microscopic simulation program `Guinea-Pig` [5] used for the current version of the parameterizations in Kíρκη simulates the passage of electrons through a bunch of electrons (and vice versa). It takes the following accelerator parameters as input:

E : the energy of the particles before the beam-beam interaction.

$N_{\mathrm{particles}}$: the number of particles per bunch.

$\epsilon_{x,y}$: the normalized horizontal and vertical emittances.

$\beta^*_{x,y}$: the horizontal and vertical beta functions.

$\sigma_{x,y,z}$: the horizontal, vertical and longitudinal beam size. A Gaussian shape is used for the charge distribution in the bunches.

f_{rep} : the repetition rate.

n_{bunch} : the number of bunches per train.

The transversal beam sizes, beta functions and normalized emittances for relativistic particles are related by

$$\beta^*_{x,y} = \frac{\sigma^2_{x,y}}{\epsilon_{x,y}} \frac{E}{m_e} \tag{2}$$

The parameters used in the most recent revision of the parameterizations are collected in tables 1 and 2. The resulting factorized electron/positron and photon distributions in version 1 of the parameterizations are depicted in figures 1 and 2.

The most important purpose of Kíρκη is to map the manifold of possible beam spectra for the NLC to a *finite* number of *reproducible* parameterizations. The distributions

$$D^{\alpha\nu\rho}_{p_1 p_2}(x_1, x_2; \sqrt{s}) \tag{3}$$

provided by Kíρκη are indexed by three integers

α : the *accelerator design class:* currently there are three options: S-band [9], Tesla [8], X-band [10, 11]. More variety will be added later, in particular the e^-e^- mode and the $e^-\gamma$ and $\gamma\gamma$ laser backscattering modes of these designs.

ν : the *version of the parameterization:* over the years, the form of the parameterizations can change, either because better approximations are found or because new simulation programs become available. All versions will remain available in order to be able to reproduce calculations.

ρ : the *revision date for the parameterization:* a particular parameterization can contain bugs, which will be fixed in subsequent revisions. While only the most recent revision should be used for new calculations, old revisions will remain available in order to be able to reproduce calculations.

The continuous parameter $\sqrt{s}$ in (3) is misleading, because accelerator parameters have been optimized for discrete values of the energy. Therefore the distributions are not available for all values of $\sqrt{s}$.

The usage of the distributions in application programs is discussed in section 3.1. Κίρκη provides for each of the distributions a non-uniform random variate generator, that generates energy fractions according to the distributions. The usage of these generators is discussed in section 3.2.

3 Usage

3.1 Distributions

A generic interface to all distributions $D_{p_1p_2}(x_1, x_2)$ is given by the `circe` function

12a ⟨*API documentation* 12a⟩≡ 13▷

```
double precision circe, d, x1, x2
integer p1, p2
d = circe (x1, x2, p1, p2)
```

where the energy fractions are specified by $x_{1,2}$ and the particles $p_{1,2}$ are identified by their standard Monte Carlo codes:[13]

12b ⟨*Particle codes* 12b⟩≡

```
integer ELECTR, POSITR, PHOTON
parameter (ELECTR =  11)
parameter (POSITR = -11)
parameter (PHOTON =  22)
```


The distributions can have integrable singularities at the end points, therefore the calling functions *must not* evaluate them at the endpoints 0 and 1. This is usually not a problem, since standard mapping techniques (cf. (10) below) will have to be used to take care of the singularity anyway. Nevertheless, all applications should favor open quadrature formulae (i.e. formulae not involving the endpoints) over closed formulae. The distributions are guaranteed to vanish unless $0 < x_{1,2} < 1$, with two exceptions. Firstly, the value -1 allows to pick up the integral of the continuum contribution:

$$D_{p_1p_2}(-1, x_2) = \lim_{\epsilon\to+0}\int_\epsilon^{1-\epsilon} dx_1\, D_{p_1p_2}(x_1, x_2) \tag{4a}$$

$$D_{p_1p_2}(x_1, -1) = \lim_{\epsilon\to+0}\int_\epsilon^{1-\epsilon} dx_2\, D_{p_1p_2}(x_1, x_2) \tag{4b}$$

$$D_{p_1p_2}(-1, -1) = \lim_{\epsilon\to+0}\int_\epsilon^{1-\epsilon} dx_1 dx_2\, D_{p_1p_2}(x_1, x_2) \tag{4c}$$

The other exception is that the strength of δ-function contributions at the endpoint can be picked up from the value at this endpoint:

$$D_{e^+e^-}(x_1, x_2) = D_{e^+e^-}(1, 1)\delta(1-x_1)\delta(1-x_2) + \text{smooth and single } \delta \tag{5a}$$

$$D_{e^\pm\gamma}(x_1, x_2) = D_{e^\pm\gamma}(1, x_2)\delta(1-x_1) + \text{smooth} \tag{5b}$$

$$D_{\gamma e^\pm}(x_1, x_2) = D_{\gamma e^\pm}(x_1, 1)\delta(1-x_2) + \text{smooth} \tag{5c}$$

The use of these special values is demonstrated in an example in section 3.1.1 below.

The distributions are normalized such that

$$\lim_{\epsilon\to+0}\int_{-\epsilon}^{1+\epsilon} dx_1 dx_2\, D_{e^+e^-}(x_1, x_2) = 1. \tag{6}$$

and the nominal e^+e^--luminosity of the currently active accelerator design can be retrieved from the database with the subroutine `circel`. The value is given in units of

$$\text{fb}^{-1}\upsilon^{-1} = 10^{32}\text{cm}^{-2}\text{sec}^{-1} \tag{7}$$

where $\upsilon = 10^7\text{sec} \approx \text{year}/\pi$ is an "effective year" of running with about 30% up-time.

13 ⟨*API documentation* 12a⟩+≡ ◁12a 14a▷

```
double precision lumi
call circel (lumi)
```

A particular parameterization is selected by the `circes` function:

14a ⟨*API documentation* 12a⟩+≡ ◁13 15▷

```
double precision x1m, x2m, roots
integer acc, ver, rev, chat
call circes (x1m, x2m, roots, acc, ver, rev, chat)
```

The parameter `roots` corresponds to the nominal center of mass energy $\sqrt{s}$/GeV of the collider. Currently $\sqrt{s} = 350\text{GeV}, 500\text{GeV}, 800\text{GeV}, 1\text{TeV}$ (i.e. `350D0`, `500D0`, `800D0` and `1000D0`) are supported. Application programs can *not* assume that energy values are interpolated. For convenience, e.g. in top threshold scans around 350GeV, a small interval around the supported values will be accepted as synonymous with the central value, but a warning will be printed. Section 5 should be consulted for the discrete values supported by a particular version of the parameterizations. Negative values of `roots` will keep the currently active value for $\sqrt{s}$.

The parameters `x1m` and `x2m` will set thresholds $x_{1,\min}$ and $x_{2,\min}$ for the event generation in the routines described in section 3.2.

The parameter `acc` selects the accelerator design. Currently the following accelerator codes are recognized:

14b ⟨*Accelerator codes* 14b⟩≡ (18b)

```
integer SBAND, TESLA, XBAND
parameter (SBAND  =  1, TESLA  =  2, XBAND  =  3)
integer SBNDEE, TESLEE, XBNDEE
parameter (SBNDEE =  4, TESLEE =  5, XBNDEE =  6)
integer NACC
parameter (NACC = 6)
```

Negative values will keep the currently active accelerator. Later I will add the e^-e^- mode and the $e^-\gamma$ and $\gamma\gamma$ laser backscattering modes of these designs:

14c ⟨*Future API documentation* 14c⟩≡

```
integer SBAND, TESLA, XBAND
parameter (SBAND  =  1, TESLA  =  2, XBAND  =  3)
integer SBNDEE, TESLEE, XBNDEE
parameter (SBNDEE =  4, TESLEE =  5, XBNDEE =  6)
integer SBNDEG, TESLEG, XBNDEG
parameter (SBNDEG =  7, TESLEG =  8, XBNDEG =  9)
integer SBNDGG, TESLGG, XBNDGG
parameter (SBNDGG = 10, TESLGG = 11, XBNDGG = 12)
integer NACC
parameter (NACC = 12)
```


The `ver` parameter is used to determine the version as follows:

`ver` > 0 : a frozen version which is documented in section 5. For example, version 1 is a family of factorized Beta distributions: $D(x_1, x_2) \propto x_1^{a_1}(1-x_1)^{b_1} x_2^{a_2}(1-x_2)^{b_2}$.

`ver` $= 0$: the latest experimental version, which is usually not documented and can change at any time without announcement.

`ver` < 0 : keep the currently active version.

The `rev` parameter is used to determine the revision of a version as follows:

`rev` > 0 : a frozen revision which is documented in section 5. The integer `rev` is constructed from the date as follows: `rev` $= 10^4 \cdot \text{year} + 10^2 \cdot \text{month} + \text{day}$, where the year is greater than 1995. Since Fortran77 ignores whitespace, it can be written like `1996 07 11` for readability. If there is no exact match, the most recent revision before the specified date is chosen.

`rev` $= 0$: the most recent revision.

`rev` < 0 : keep the currently active revision.

Finally, the parameter `chat` controls the "chattiness" of `circe`. If it is 0, only error messages are printed. If it is 1, the parameters in use are printed whenever they change. Higher values of `chat` can produce even more diagnostics.

In addition to the generic interface `circe`, there are specialized functions for particular particle distributions. Obviously

$$D^{\alpha\nu\rho}_{e^\pm\gamma}(x_1, x_2, s) = D^{\alpha\nu\rho}_{\gamma e^\pm}(x_2, x_1, s) \tag{8}$$

and there are three independent functions $D_{e^-e^+}$, $D_{e^-\gamma}$ and $D_{\gamma\gamma}$ for the e^+e^- colliders with reasonable mnemonics:

15 ⟨*API documentation* 12a⟩+≡ ◁14a 22a▷

```
double precision circee, circeg, circgg
d = circee (x1, x2)
d = circeg (x1, x2)
d = circgg (x1, x2)
```

Calling the latter three functions is marginally faster in the current implementation, but this can change in the future.

3.1.1 Example

For clarification, let me give a simple example. Imagine we want to calculate the integrated production cross section

$$\sigma_X(s) = \int dx_1\, dx_2\, \sigma_{e^+e^-\to X}(x_1 x_2 s) D_{e^+e^-}(x_1, x_2, s) \tag{9}$$

Since the distributions are singular in the $x_{1,2} \to 1$ limit, we have to map away this singularity with

$$x \to t = (1-x)^{1/\eta} \tag{10a}$$

Therefore

$$\int_0^1 dx\, f(x) = \int_0^1 dt\, \eta t^{\eta-1} f(1-t^\eta) \tag{10b}$$

with η sufficiently large to give the integrand a finite limit at $x \to 1$. If f diverges like a power $f(x) \propto 1/(1-x)^\beta$, this means $\eta > 1/(1-\beta)$.

As a specific example, let us "measure" a one particle s-channel exchange cross section

$$\sigma(s) \propto \frac{1}{s} \tag{11}$$

16a ⟨sample.f 16a⟩≡ 17a ▷

```
      double precision function sigma (s)
      implicit none
      double precision s
      sigma = 1d0 / s
      end
```

I will present the example code in a bottom-up fashion, which should be intuitive and is described in some more detail in appendix A. Assuming the existence of a one- and a two-dimensional Gaussian integration function `gauss1` and `gauss2`,[1] we can perform the integral as follows:

16b ⟨*Gauss integration* 16b⟩≡ (18b)

```
      s = sigma (1d0) * circee (1d0, 1d0)
     $   + gauss1 (d1, 0d0, 1d0, EPS)
     $   + gauss1 (d2, 0d0, 1d0, EPS)
     $   + gauss2 (d12, 0d0, 1d0, 0d0, 1d0, EPS)
      write (*, 1000) 'delta(sigma) (Gauss) =', (s-1d0)*100d0
1000  format (1X, A22, 1X, F5.2, '%')
```

Note how the four combinations of continuum and δ-peak are integrated separately, where you have to use three auxiliary functions `d1`, `d2` and `d12`. The continuum contribution, including the Jacobian:

17a ⟨sample.f 16a⟩+≡ ◁16a 17b▷

```
      double precision function d12 (t1, t2)
      implicit none
      double precision t1, t2, x1, x2, sigma, circee
      ⟨EPS & PWR 18a⟩
      x1 = 1d0 - t1**PWR
      x2 = 1d0 - t2**PWR
      d12 = PWR*PWR * (t1*t2)**(PWR-1d0)
     $      * sigma (x1*x2) * circee (x1, x2)
      end
```

the first product of continuum and δ-peak:

17b ⟨sample.f 16a⟩+≡ ◁17a 17c▷

```
      double precision function d1 (t1)
      implicit none
      double precision t1, x1, sigma, circee
      ⟨EPS & PWR 18a⟩
      x1 = 1d0 - t1**PWR
      d1 = PWR * t1**(PWR-1d0) * sigma (x1) * circee (x1, 1d0)
      end
```

and the second one:

17c ⟨sample.f 16a⟩+≡ ◁17b 18b▷

```
      double precision function d2 (t2)
      implicit none
      double precision t2, x2, sigma, circee
      ⟨EPS & PWR 18a⟩
      x2 = 1d0 - t2**PWR
      d2 = PWR * t2**(PWR-1d0) * sigma (x2) * circee (1d0, x2)
      end
```

Below you will see that the power of the singularity of the e^+e^- distributions at $x \to 1$ is $\approx -2/3$. To be on the safe side, we choose the power η in (10) as 5. It is kept in

[1] They are provided in the example program `sample.f`.

the parameter PWR, while EPS is the desired accuracy of the Gaussian integration:

18a ⟨EPS & PWR 18a⟩≡ (17 18b)

```
double precision EPS, PWR
parameter (EPS = 1d-6, PWR = 5d0)
```

These code fragments can now be used in a main program that loops over energies and accelerator designs

18b ⟨sample.f 16a⟩+≡ ◁17c 23b▷

```
program sample
implicit none
⟨Accelerator codes 14b⟩
⟨EPS & PWR 18a⟩
⟨Other variables in sample 21⟩
integer acc, ver, i
double precision roots(5)
data roots / 350D0, 500D0, 800D0, 1000D0, 1600D0 /
ccc   do 10 acc = 1, NACC
      do 10 acc = 1, 3
         do 11 ver = 1, 5
            if (ver .eq. 2) goto 13
            do 12 i = 1, 5
               if ((ver .eq. 3) .or. (ver .eq. 4)) then
                  if ((acc .ne. TESLA) .or. (roots(i) .ne. 800d0)) then
                     goto 14
                  endif
               else if (ver .eq. 5) then
                  if ((acc .ne. TESLA)
     $                 .or. (roots(i) .lt. 450d0)
     $                 .or. (roots(i) .gt. 850d0)) then
                     goto 14
                  endif
               endif
               call circes (0d0, 0d0, roots(i), acc, ver, 19980505, 1)
               ⟨Gauss integration 16b⟩
               ⟨Monte Carlo integration 22d⟩
14             continue
12          continue
13          continue
11       continue
10    continue
      end
```

with the following result

19 ⟨Sample output 19⟩≡

```
circe:message: starting up ...
circe:message: $Id: prelude.nw,v 1.38 1998/05/05 10:37:33 ohl Exp $
circe:message: updating 'roots' to  350.0
circe:message: updating 'acc' to SBAND
circe:message: updating 'ver' to  1
circe:message: updating 'rev' to 19980505
delta(sigma) (Gauss) =  1.64%
delta(sigma) (MC)    =  1.64%
                     +/-  0.03%
circe:message: updating 'roots' to  500.0
delta(sigma) (Gauss) =  3.22%
delta(sigma) (MC)    =  3.22%
                     +/-  0.06%
circe:message: updating 'roots' to  800.0
delta(sigma) (Gauss) =  7.27%
delta(sigma) (MC)    =  7.24%
                     +/-  0.14%
circe:message: updating 'roots' to 1000.0
delta(sigma) (Gauss) = 10.58%
delta(sigma) (MC)    = 10.12%
                     +/-  0.20%
circe:message: updating 'roots' to 1600.0
delta(sigma) (Gauss) = 10.58%
delta(sigma) (MC)    = 10.39%
                     +/-  0.20%
circe:message: updating 'roots' to  350.0
circe:message: updating 'acc' to TESLA
delta(sigma) (Gauss) =  2.74%
delta(sigma) (MC)    =  2.72%
                     +/-  0.05%
circe:message: updating 'roots' to  500.0
delta(sigma) (Gauss) =  2.89%
```

```
delta(sigma) (MC)    =  2.83%
                    +/-  0.05%
circe:message: updating 'roots' to  800.0
delta(sigma) (Gauss) =  6.92%
delta(sigma) (MC)    =  6.81%
                    +/-  0.11%
circe:message: updating 'roots' to 1000.0
delta(sigma) (Gauss) = 10.27%
delta(sigma) (MC)    = 10.15%
                    +/-  0.17%
circe:message: updating 'roots' to 1600.0
delta(sigma) (Gauss) =  7.84%
delta(sigma) (MC)    =  7.79%
                    +/-  0.16%
circe:message: updating 'roots' to  800.0
circe:message: updating 'ver' to  3
circe:warning: version 3 retired after 97/04/17
circe:message: falling back to version 4
delta(sigma) (Gauss) =  2.24%
delta(sigma) (MC)    =  2.25%
                    +/-  0.05%
circe:message: updating 'ver' to  4
delta(sigma) (Gauss) =  2.24%
delta(sigma) (MC)    =  2.25%
                    +/-  0.05%
circe:message: updating 'roots' to  500.0
circe:message: updating 'ver' to  5
delta(sigma) (Gauss) =  3.06%
delta(sigma) (MC)    =  3.03%
                    +/-  0.06%
circe:message: updating 'roots' to  800.0
delta(sigma) (Gauss) =  6.92%
delta(sigma) (MC)    =  6.83%
                    +/-  0.11%
circe:message: updating 'roots' to  350.0
circe:message: updating 'acc' to XBAND
circe:message: updating 'ver' to  1
```



```
delta(sigma) (Gauss) =  2.64%
delta(sigma) (MC)    =  2.64%
                    +/-  0.06%
circe:message: updating 'roots' to  500.0
delta(sigma) (Gauss) =  3.58%
delta(sigma) (MC)    =  3.63%
                    +/-  0.09%
circe:message: updating 'roots' to  800.0
delta(sigma) (Gauss) = 13.32%
delta(sigma) (MC)    = 13.18%
                    +/-  0.31%
circe:message: updating 'roots' to 1000.0
delta(sigma) (Gauss) = 15.21%
delta(sigma) (MC)    = 15.50%
                    +/-  0.39%
circe:message: updating 'roots' to 1600.0
delta(sigma) (Gauss) = 21.87%
delta(sigma) (MC)    = 22.40%
                    +/-  0.91%
```

We almost forgot to declare the variables in the main program

21 ⟨*Other variables in* `sample` 21⟩≡ (18b) 23a▷

```
double precision s
double precision gauss1, gauss2, circee, sigma, d1, d2, d12
external d1, d2, d12
```

This concludes the integration example. It should have made it obvious how to proceed in a realistic application.

In section 3.2.1 below, I will describe a Monte Carlo method for calculating such integrals efficiently.

3.2 Generators

The function `circe` and its companions are opaque to the user. Since they will in general contain singularities, applications will *not* be able to generate corresponding samples of random numbers efficiently. To fill this gap, four random number generators are provided. The subroutine `girce` will generate particle types $p_{1,2}$ and

energy fractions $x_{1,2}$ in one step, according to the selected distribution.[2] Particle p_1 will be either a positron or a photon and p_2 will be either an electron or a photon. The energy fractions are guaranteed to be above the currently active thresholds: $x_i \geq x_{i,\min}$. This can be used to cut on soft events—the photon distributions are rather soft—which might not be interesting in most simulations.

22a ⟨*API documentation* 12a⟩+≡ ◁15 22b▷

```
call girce  (x1, x2, p1, p2, rng)
```

The output parameters of `girce` are identical to the input parameters of `circe`, with the exception of `rng`. The latter is a subroutine with a single double precision argument, which will be assigned a uniform deviate from the interval $[0, 1]$ after each call:

22b ⟨*API documentation* 12a⟩+≡ ◁22a 22c▷

```
subroutine rng (r)
double precision r
r = ⟨uniform deviate on [0, 1] (never defined)⟩
end
```

Typically, it will be just a wrapper around the standard random number generator of the application program. For studies with a definite initial state, three generator functions are available.

22c ⟨*API documentation* 12a⟩+≡ ◁22b

```
call gircee (x1, x2, rng)
call girceg (x1, x2, rng)
call gircgg (x1, x2, rng)
```

3.2.1 Example

Returning to the example from section 3.2.1, I present a concise Monte Carlo algorithm for calculating the same integral:

22d ⟨*Monte Carlo integration* 22d⟩≡ (18b)

```
s = 0d0
s2 = 0d0
do 100 n = 1, NEVENT
   call gircee (x1, x2, random)
   w = sigma (x1*x2)
   s = s + w
   s2 = s2 + w*w
100 continue
s = s / dble(NEVENT)
s2 = s2 / dble(NEVENT)
write (*, 1000) 'delta(sigma) (MC)    =', (s-1d0)*100d0
write (*, 1000) '                    +/-',
$               sqrt((s2-s*s)/dble(NEVENT))*100d0
```

[2] The implementation of the flavor selection with non-vanishing thresholds $x_{1,\min}$ and $x_{2,\min}$ is moderately inefficient at the moment. It can be improved by a factor of two.

23a ⟨*Other variables in* `sample` 21⟩+≡ (18b) ◁21

```
double precision w, s2, x1, x2
external random
integer NEVENT, n
parameter (NEVENT = 10000)
```

Here is a simple linear congruential random number generator for the sample program. Real applications will use their more sophisticated generators instead.

23b ⟨`sample.f` 16a⟩+≡ ◁18b

```
subroutine random (r)
implicit none
double precision r
integer m, a, c
parameter (M = 259200, A = 7141, C = 54773)
integer n
save n
data n /0/
n = mod(n*a+c,m)
r = dble (n) / dble (m)
end
```

If the cross section is slowly varying on the range where the $x_{1,2}$ distributions are non-zero, this algorithm is very efficient.

However, if this condition is not met, the explicit form of the parameterizations in section 5 should be consulted and appropriate mapping techniques should be applied. The typical example for this problem is a narrow resonance just below the nominal beam energy.

3.2.2 Event Generators

For Monte Carlo event generators that use the standard `/hepevt/` common block [14], the addition of the Kíρχη library is trivial. During the initialization of the event generator, the `circes` subroutine is called to set up Kíρχη's internal state. For example:

24a ⟨*Initialize event generator* 24a⟩≡

```
call circes (0d0, 0d0, roots, acc, ver, 1996 07 11, 1)
```

During event generation, before setting up the e^+e^- initial state, the `gircee` subroutine is called with the event generator's random number generator:

24b ⟨*Event generation* 24b⟩≡ 24c ▷

```
call gircee (x1, x2, random)
```

The resulting energy fractions x_1 and x_2 are now available for defining the initial state electron

24c ⟨*Event generation* 24b⟩+≡ ◁ 24b 24d ▷

```
isthep(1) = 101
idhep(1) = ELECTR
phep(1,1) = 0d0
phep(2,1) = 0d0
phep(3,1) = x1 * ebeam
phep(4,1) = x1 * ebeam
phep(5,1) = 0d0
```

and positron.

24d ⟨*Event generation* 24b⟩+≡ ◁ 24c

```
isthep(2) = 102
idhep(2) = POSITR
phep(1,2) = 0d0
phep(2,2) = 0d0
phep(3,2) = - x2 * ebeam
phep(4,2) = x2 * ebeam
phep(5,2) = 0d0
```

Using Kíρχη with other event generators should be straightforward as well.

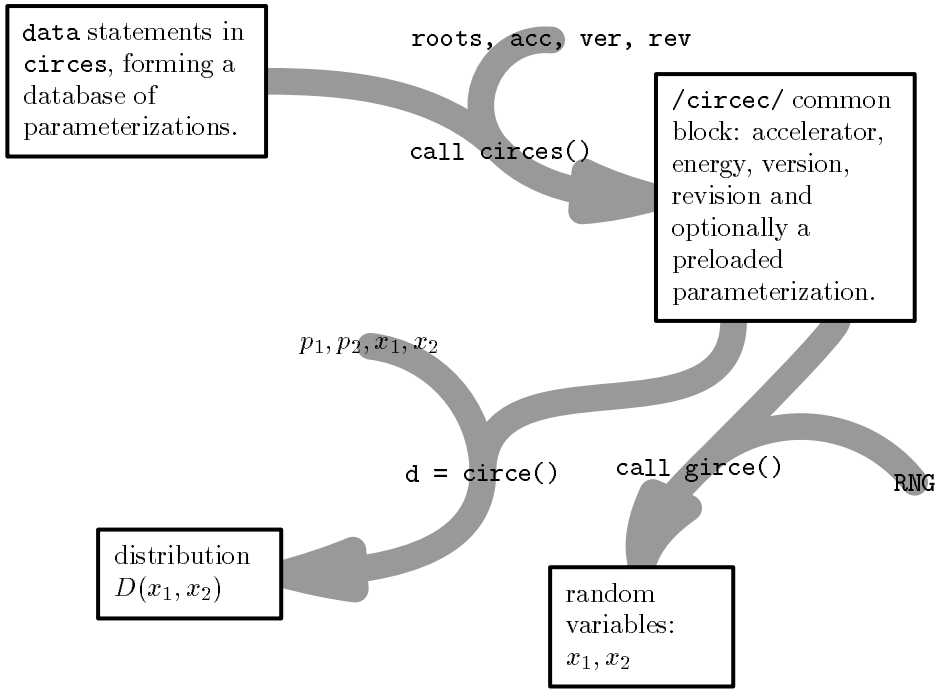

Figure 5: Architecture of Kíρχη: `circes()` selects energy and accelerator and loads the parameterization. The function `circe()` calculates the values of the selected distribution function at the given energy fractions. The subroutine `girce()` generates energy fractions using a specified random number generator in accordance with the selected distribution.

4 Technical Notes

The structure of Κίρκηη is extremely simple (cf. figure 5) and is mainly a bookkeeping excercise. All that needs to be done is to maintain a database of available parameterizations and to evaluate the corresponding functions. The only non trivial algorithms are used for the efficient generation of random deviates.

I have avoided the use of initialized `common` blocks (i.e. `block data` subroutines), because the Fortran77 standard does not provide a *portable* way of ensuring that `block data` subroutines are actually executed at loading time. Instead, the `/circom/` common block is tagged by a "magic number" to check for initialization and its members are filled by the `circes` subroutine when necessary.

A more flexible method would be to replace the `data` statements by reading external files. This option causes portability problems, however, because I would have to make sure that the names of the external files are valid in all files systems of the target operating systems. More significantly, splitting the implementation into several parts forces the user to keep all files up to date. This can be a problem, because Fortran source files and data input files will typically be kept in different parts of the file system.

The option of implementing Κίρκηη statelessly, i.e. with pure function calls and without `common` blocks, has been dismissed. While it would have been more straightforward on the side of the library, it would have placed the burdon of maintaining state (accelerator, energy, etc.) on the application program, thereby complicating them considerably. Keeping an explicit state in Κίρκηη has the additional benefit of allowing to precompute certain internal variables, resulting in a more efficient implementation.

5 Parameterizations

Version 1.02β of Κίρκηη supports just one version of the parameterizations. Future versions will provide additional parameterizations.

5.1 Version 1

The first version of the parameterization uses a simple factorized *ansatz*

$$D^{\alpha_1\rho}_{p_1p_2}(x_1,x_2,s) = d^{\alpha_1\rho}_{p_1}(x_1)d^{\alpha_1\rho}_{p_2}(x_2) \tag{12a}$$

	SBAND	TESLA	TESLA'	XBAND
$\mathcal{L}/\mathrm{fb}^{-1}\upsilon^{-1}$	$31.38^{+0.22}_{-0.22}$	$106.25^{+0.71}_{-0.71}$	$95.24^{+0.73}_{-0.73}$	$36.39^{+0.29}_{-0.29}$
$\int d_{e\pm}$	$0.4812^{+0.0041}_{-0.0041}$	$0.5723^{+0.0046}_{-0.0045}$	$0.3512^{+0.0048}_{-0.0048}$	$0.3487^{+0.0040}_{-0.0040}$
$x^{\alpha}_{e\pm}$	$11.1534^{+0.0770}_{-0.0761}$	$15.2837^{+0.0923}_{-0.0914}$	$27.1032^{+0.3071}_{-0.3019}$	$6.9853^{+0.0733}_{-0.0718}$
$(1-x_{e\pm})^{\alpha}$	$-0.6302^{+0.0013}_{-0.0012}$	$-0.6166^{+0.0011}_{-0.0011}$	$-0.6453^{+0.0017}_{-0.0017}$	$-0.6444^{+0.0017}_{-0.0017}$
$\int d_{\gamma}$	$0.6237^{+0.0033}_{-0.0033}$	$0.7381^{+0.0036}_{-0.0036}$	$0.3502^{+0.0034}_{-0.0034}$	$0.4149^{+0.0031}_{-0.0031}$
x^{α}_{γ}	$-0.6911^{+0.0006}_{-0.0006}$	$-0.6921^{+0.0006}_{-0.0006}$	$-0.6947^{+0.0011}_{-0.0011}$	$-0.6876^{+0.0010}_{-0.0010}$
$(1-x_{\gamma})^{\alpha}$	$14.9355^{+0.0761}_{-0.0754}$	$24.1647^{+0.1124}_{-0.1116}$	$33.6576^{+0.3021}_{-0.2983}$	$8.3227^{+0.0659}_{-0.0649}$

Table 5: Version 1, revision 1997 04 16 of the beam spectra at 500 GeV. The rows correspond to the luminosity per effective year, the integral over the continuum and the powers in the factorized Beta distributions (12).

	SBAND	TESLA	TESLA'	XBAND
$\mathcal{L}/\mathrm{fb}^{-1}\upsilon^{-1}$	$119.00^{+0.83}_{-0.83}$	214.33^{+0***}_{-0***}	212.22^{+0***}_{-0***}	$118.99^{+0.91}_{-0.91}$
$\int d_{e\pm}$	$0.5604^{+0.0040}_{-0.0039}$	$0.6686^{+0.0040}_{-0.0040}$	$0.4448^{+0.0043}_{-0.0043}$	$0.5001^{+0.0038}_{-0.0038}$
$x^{\alpha}_{e\pm}$	$4.2170^{+0.0258}_{-0.0255}$	$5.5438^{+0.0241}_{-0.0239}$	$9.6341^{+0.0814}_{-0.0803}$	$2.6184^{+0.0192}_{-0.0190}$
$(1-x_{e\pm})^{\alpha}$	$-0.6118^{+0.0013}_{-0.0013}$	$-0.5847^{+0.0011}_{-0.0011}$	$-0.6359^{+0.0014}_{-0.0014}$	$-0.6158^{+0.0015}_{-0.0015}$
$\int d_{\gamma}$	$0.7455^{+0.0032}_{-0.0032}$	$1.0112^{+0.0033}_{-0.0033}$	$0.4771^{+0.0031}_{-0.0031}$	$0.6741^{+0.0031}_{-0.0031}$
x^{α}_{γ}	$-0.6870^{+0.0006}_{-0.0006}$	$-0.6908^{+0.0004}_{-0.0004}$	$-0.6936^{+0.0008}_{-0.0008}$	$-0.6834^{+0.0007}_{-0.0007}$
$(1-x_{\gamma})^{\alpha}$	$6.7145^{+0.0310}_{-0.0308}$	$9.9992^{+0.0342}_{-0.0340}$	$13.1607^{+0.0896}_{-0.0886}$	$3.8589^{+0.0215}_{-0.0213}$

Table 6: Version 1, revision 1997 04 17 of the beam spectra at 1 TeV.

	350 GeV	500 GeV	800 GeV	1600 GeV
$\mathcal{L}/\mathrm{fb}^{-1}\upsilon^{-1}$	$97.45^{+0.67}_{-0.67}$	$106.25^{+0.71}_{-0.71}$	170.86^{+0***}_{-0***}	340.86^{+0***}_{-0***}
$\int d_{e\pm}$	$0.6093^{+0.0049}_{-0.0049}$	$0.5723^{+0.0046}_{-0.0045}$	$0.6398^{+0.0042}_{-0.0041}$	$0.5094^{+0.0040}_{-0.0040}$
$x^{\alpha}_{e\pm}$	$17.6137^{+0.1065}_{-0.1055}$	$15.2837^{+0.0923}_{-0.0914}$	$7.6221^{+0.0365}_{-0.0361}$	$5.0550^{+0.0353}_{-0.0349}$
$(1-x_{e\pm})^{\alpha}$	$-0.6061^{+0.0011}_{-0.0011}$	$-0.6166^{+0.0011}_{-0.0011}$	$-0.5944^{+0.0011}_{-0.0011}$	$-0.6187^{+0.0013}_{-0.0013}$
$\int d_{\gamma}$	$0.7729^{+0.0039}_{-0.0039}$	$0.7381^{+0.0036}_{-0.0036}$	$0.9178^{+0.0034}_{-0.0034}$	$0.5875^{+0.0031}_{-0.0031}$
x^{α}_{γ}	$-0.6949^{+0.0006}_{-0.0006}$	$-0.6921^{+0.0006}_{-0.0006}$	$-0.6908^{+0.0005}_{-0.0005}$	$-0.6892^{+0.0007}_{-0.0007}$
$(1-x_{\gamma})^{\alpha}$	$28.9399^{+0.1370}_{-0.1361}$	$24.1647^{+0.1124}_{-0.1116}$	$13.1167^{+0.0497}_{-0.0495}$	$7.5514^{+0.0428}_{-0.0424}$

Table 7: Version 1, revision 1997 04 17 of the beam spectra for TESLA.

	500 GeV	800 GeV
$\mathcal{L}/\mathrm{fb}^{-1}\upsilon^{-1}$	$339.80^{+0.83}_{-0.83}$	$359.36^{+0.93}_{-0.93}$
$\int d_{e\pm}$	$0.5019^{+0.0016}_{-0.0016}$	$0.4125^{+0.0016}_{-0.0016}$
$x^{\alpha}_{e\pm}$	$12.2867^{+0.0318}_{-0.0316}$	$13.3242^{+0.0442}_{-0.0440}$
$(1-x_{e\pm})^{\alpha}$	$-0.6276^{+0.0005}_{-0.0005}$	$-0.6401^{+0.0005}_{-0.0005}$
$\int d_{\gamma}$	$0.5114^{+0.0012}_{-0.0012}$	$0.3708^{+0.0011}_{-0.0011}$
x^{α}_{γ}	$-0.6912^{+0.0003}_{-0.0003}$	$-0.6924^{+0.0004}_{-0.0004}$
$(1-x_{\gamma})^{\alpha}$	$17.0673^{+0.0375}_{-0.0375}$	$16.8145^{+0.0482}_{-0.0480}$

Table 8: Version 5, revision 1998 05 05 of the beam spectra for high luminosity TESLA.

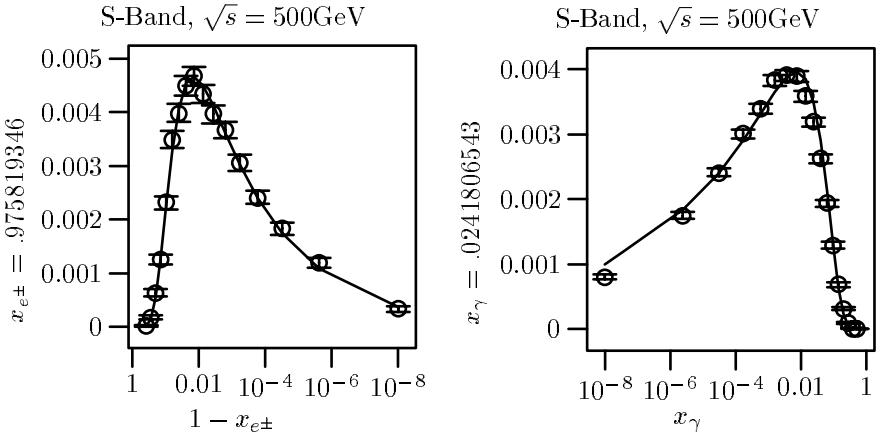

Figure 6: Fit of the $e^{\pm}$- and γ-distributions for the S-Band design at $\sqrt{s} = 500$GeV. The open circles with error bars are the result of the `Guinea-Pig` similation. The full line is the fit.

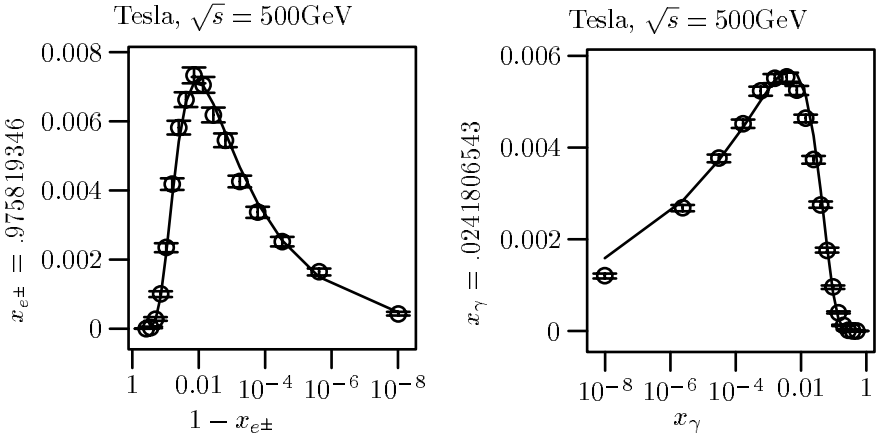

Figure 7: Fit of the $e^{\pm}$- and γ-distributions for the Tesla design at $\sqrt{s} = 500$GeV.

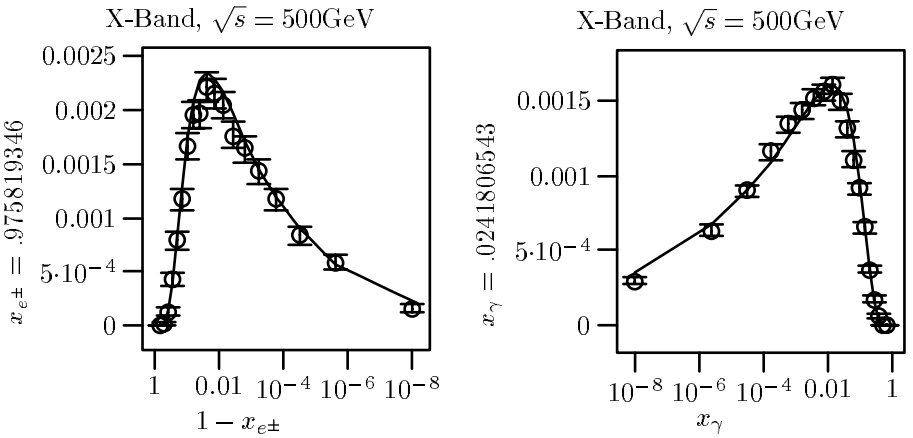

Figure 8: Fit of the $e^\pm$- and γ-distributions for the X-Band design at $\sqrt{s} = 500$GeV.

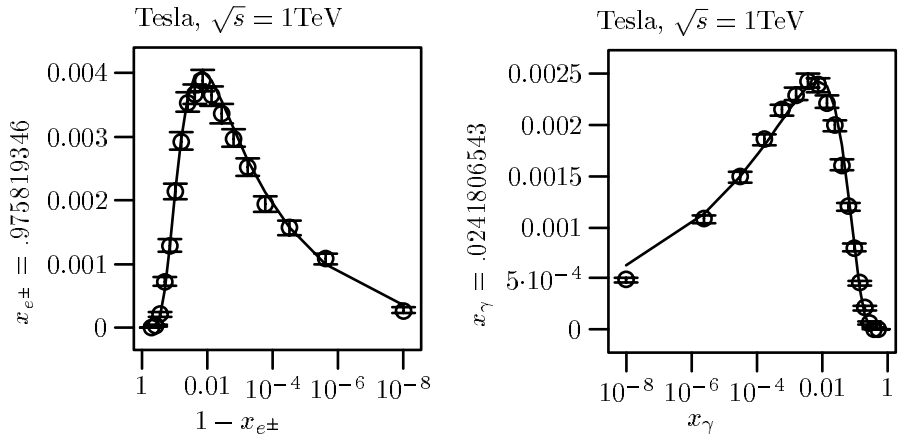

Figure 9: Fit of the $e^\pm$- and γ-distributions for the Tesla design at $\sqrt{s} = 1$TeV.

	SBNDEE	TESLEE	XBNDEE
$\mathcal{L}/\mathrm{fb}^{-1}\upsilon^{-1}$	$9.29^{+0.06}_{-0.06}$	$21.62^{+0.17}_{-0.17}$	$13.97^{+0.10}_{-0.10}$
$\int d_{e\pm}$	$.6513^{+0.0059}_{-0.0059}$	$.7282^{+0.0083}_{-0.0082}$	$.5270^{+0.0049}_{-0.0049}$
$x^\alpha_{e\pm}$	$10.3040^{+0.0601}_{-0.0593}$	$14.8578^{+0.1047}_{-0.1034}$	$5.8897^{+0.0455}_{-0.0448}$
$(1-x_{e\pm})^\alpha$	$-.5946^{+0.0015}_{-0.0015}$	$-.5842^{+0.0018}_{-0.0018}$	$-.6169^{+0.0016}_{-0.0015}$
$\int d_\gamma$	$.4727^{+0.0035}_{-0.0035}$	$.5300^{+0.0046}_{-0.0046}$	$.3746^{+0.0029}_{-0.0029}$
x^α_γ	$-.6974^{+0.0009}_{-0.0009}$	$-.7039^{+0.0009}_{-0.0009}$	$-.6892^{+0.0010}_{-0.0010}$
$(1-x_\gamma)^\alpha$	$20.6447^{+0.1513}_{-0.1497}$	$36.1286^{+0.3027}_{-0.2991}$	$10.0872^{+0.0822}_{-0.0815}$

Table 9: *Experimental* Version 1, revision 0 of the beam spectra at 500 GeV. The rows correspond to the luminosity per effective year, the integral over the continuum and the powers in the factorized Beta distributions (12).

where the distributions are simple Beta distributions:

$$d^{\alpha 1\rho}_{e^\pm}(x) = a_0^{\alpha\rho}\delta(1-x) + a_1^{\alpha\rho}x^{a_2^{\alpha\rho}}(1-x)^{a_3^{\alpha\rho}} \tag{12b}$$

$$d^{\alpha 1\rho}_{\gamma}(x) = a_4^{\alpha\rho}x^{a_5^{\alpha\rho}}(1-x)^{a_6^{\alpha\rho}} \tag{12c}$$

This form of the distributions is motivated by the observation [2] that the $e^\pm$ distributions diverge like a power for $x \to 1$ and vanish at $x \to 0$. The behavior of the γ distributions is similar with the borders exchanged.

5.1.1 Fitting

The parameters a_i in (12) have been obtained by a least-square fit of (12) to histograms of simulation results from `Guinea-Pig`. Some care has to taken when fitting singular distributions to histogrammed data. Obviously equidistant bins are not a good idea, because most bins will be almost empty (cf. figures 1 and 2) and consequently a lot of information will be wasted. One solution to this problem is the use of logarithmic bins. This, however, maps the compact region $[0,1]\times[0,1]$ to $[-\infty,0]\times[-\infty,0]$, which is inconvenient because of the missing lower bounds.

The more appropriate solution is to use two maps

$$\begin{aligned}\phi : [0,1] &\to [0,1]\\ x &\mapsto y = x^{1/\eta}\end{aligned} \tag{13}$$

	SBNDEE	TESLEE	XBNDEE
$\mathcal{L}/\mathrm{fb}^{-1}\upsilon^{-1}$	$45.59^{+0.34}_{-0.34}$	$25.47^{+0.20}_{-0.20}$	$41.06^{+0.28}_{-0.28}$
$\int d_{e^\pm}$	$.7892^{+0.0075}_{-0.0074}$	$.6271^{+0.0066}_{-0.0065}$	$.7203^{+0.0058}_{-0.0058}$
$x_{e^\pm}^\alpha$	$5.4407^{+0.0285}_{-0.0281}$	$8.7504^{+0.0669}_{-0.0658}$	$2.7415^{+0.0121}_{-0.0119}$
$(1-x_{e^\pm})^\alpha$	$-.5285^{+0.0020}_{-0.0020}$	$-.6058^{+0.0017}_{-0.0017}$	$-.5049^{+0.0020}_{-0.0020}$
$\int d_\gamma$	$.6403^{+0.0040}_{-0.0040}$	$.4278^{+0.0038}_{-0.0038}$	$.6222^{+0.0032}_{-0.0032}$
x_γ^α	$-.6960^{+0.0008}_{-0.0008}$	$-.6982^{+0.0010}_{-0.0010}$	$-.6795^{+0.0008}_{-0.0008}$
$(1-x_\gamma)^\alpha$	$12.4803^{+0.0839}_{-0.0831}$	$18.5260^{+0.1674}_{-0.1655}$	$4.7506^{+0.0262}_{-0.0260}$

Table 10: *Experimental* Version 1, revision 0 of the beam spectra at 1 TeV.

	350 GeV	500 GeV	800 GeV
$\mathcal{L}/\mathrm{fb}^{-1}\upsilon^{-1}$	$15.18^{+0.13}_{-0.13}$	$21.62^{+0.17}_{-0.17}$	$43.98^{+0.38}_{-0.38}$
$\int d_{e^\pm}$	$.6691^{+0.0083}_{-0.0083}$	$.7282^{+0.0083}_{-0.0082}$	$.7701^{+0.0090}_{-0.0089}$
$x_{e^\pm}^\alpha$	$25.2753^{+0.2040}_{-0.2007}$	$14.8578^{+0.1047}_{-0.1034}$	$8.1905^{+0.0543}_{-0.0535}$
$(1-x_{e^\pm})^\alpha$	$-.5994^{+0.0017}_{-0.0017}$	$-.5842^{+0.0018}_{-0.0018}$	$-.5575^{+0.0021}_{-0.0021}$
$\int d_\gamma$	$.4464^{+0.0047}_{-0.0047}$	$.5300^{+0.0046}_{-0.0046}$	$.5839^{+0.0047}_{-0.0047}$
x_γ^α	$-.7040^{+0.0011}_{-0.0011}$	$-.7039^{+0.0009}_{-0.0009}$	$-.7046^{+0.0009}_{-0.0009}$
$(1-x_\gamma)^\alpha$	$60.1882^{+0.5882}_{-0.5797}$	$36.1286^{+0.3027}_{-0.2991}$	$19.3944^{+0.1681}_{-0.1660}$

Table 11: *Experimental* Version 1, revision 0 of the beam spectra for `TESLEE`.

where $x = x_\gamma$ or $x = 1 - x_{e^\pm}$, and to bin the result equidistantly. If η is chosen properly (cf. (10)), the bin contents will then fall off at the singularity. The fits in tables 5, 6, and 7 have been performed with $\eta = 5$ and the resulting bin contents can be read off from figures 6–9.

Using this procedure for binning the results of the simulations, the popular fitting package `MINUIT` [15] converges quickly in all cases considered. The resulting parameters are given in tables 5, 6, and 7. Plots of the corresponding distributions have been shown in figures 1 and 2. It is obvious that an *ansatz* like (12) is able to distinguish among the accelerator designs. Thus it can provide a solid basis for physics studies.

In figures 6–9 I give a graphical impression of the quality of the fit, which appears to be as good as one could reasonably expect for a simple *ansatz* like (12). Note that the histograms have non-equidistant bins and that the resulting Jacobians have not been removed. Therefore the bin contents falls off at the singularities, as discussed above.

The errors used for the least-square fit had to be taken from a Monte Carlo (MC) study. `Guinea-Pig` only provides the $\sqrt{n}$ from Poissonian statistics for each bin, but the error accumulation during tracking the particles through phase space is not available. The MC studies shows that the latter error dominates the former, but appears to be reasonably Gaussian. A complete MC study of all parameter sets is computationally expensive (more than a week of processor time on a fast SGI). From an exemplary MC study of a few parameter sets, it appears that the errors can be described reasonably well by rescaling the Poissonian error in each bin with appropriate factors for electrons/positrons and photons and for continuum and delta. This procedure has been adopted.

The χ^2/d.o.f.'s of the fits are less than $\mathcal{O}(10)$. The simple *ansatz* (12) is therefore very satisfactory. In fact, trying to improve the ad-hoc factorized Beta distributions by the better motivated approximations from [7] or [16], it turns out [17] that (12) provides a significantly better fit of the results of the simulations. The price to pay is that the parameters in (12) have no direct physical interpretation.

5.1.2 Generators

For this version of the parameterizations we need a fast generator of Beta distributions:

$$\beta^{a,b}(x) \propto x^{a-1}(1-x)^{b-1} \tag{14}$$

This problem has been studied extensively and we can use a published algorithm [18] that is guaranteed to be very fast for all a, b such that $0 < a \leq 1 \leq b$, which turns

out to be always the case (cf. tables 5, 6, and 7).

5.2 Future Versions

There are two ways in which the parameterizations can be improved:

more complicated functions: the factorized fits can only be improved marginally by adding more positive semi-definite factors to (12). More improvement is possible by using sums of functions, but in this case, the best fits violate the positivity requirement and have to be discarded.

correlations: the parameterization in section 5.1 is factorized. While this is a good approximation, the simulations nevertheless show correlations among x_1 and x_2. These correlations can be included in a future version.

interpolation: the parameterization in section 5.1 is based on fitting the simulation results by simple functions. Again, this appears to be a good approximation. But such fits can not uncover any fine structure of the distributions. Therefore it will be worthwhile to study interpolations of the simulation results in the future. A proper interpolation of results with statistical errors is however far from trivial: straightforward polynomial or spline interpolations will be oscillatory and violate the positivity requirement. Smoothing algorithms have to be investigated in depth before such a parameterization can be released.

other simulations: besides [5], other simulation codes are invited to contribute their results for inclusion in the Κίρκη library.

6 Conclusions

I have presented a library of simple parameterizations of realistic $e^{\pm}$- and γ-beam spectra at future linear e^+e^--colliders. The library can be used for integration and event generation. Emphasis is put on simplicity and reproducibility of the parameterizations for supporting reproducible physics simulations.

Acknowledgements

Daniel Schulte made his simulation code `Guinea-Pig` available and answered questions. Harald Anlauf and Torbjörn Sjöstrand have contributed useful suggestions. The Tesla group at DESY/Zeuthen made error estimates feasible by donating time on the multi-headed number cruncher ῾Υδρα. The 1996 ECFA/Desy Linear Collider Workshop got me started and provided support. Thanks to all of them.

References

[1] H. Murayama and M. E. Peskin, SLAC-PUB-7149, to appear in *Ann. Rev. Nucl. Part. Sci.*; P. Zerwas, DESY 94-001-REV.

[2] P. Chen and R. J. Noble, SLAC-PUB-4050; M. Bell and J. S. Bell, Part. Accl. **24**, 1 (1988); R. Blankenbecler and S. D. Drell, Phys. Rev. Lett. **61**, 2324 (1988); P. Chen and K. Yokoya, Phys. Rev. Lett. **61**, 1101 (1988); M. Jacob and T. T. Wu, Nucl. Phys. **B303**, 389 (1988); V. N. Baier, V. M. Katkov, and V. M. Strakovenkov, Nucl. Phys. **B328**, 387 (1989); R. Blankenbecler, S. D. Drell, and N. Kroll, Phys. Rev. **D40**, 2462 (1989); P. Chen and V. L. Telnov, Phys. Rev. Lett. **63**, 1796 (1989).

[3] K. Yokoya, KEK 85-9, KEK.

[4] P. Chen *et al.*, Nucl. Inst. Meth. **A355**, 107 (1995).

[5] D. Schulte, Ph.D. thesis, in preparation.

[6] R. B. Palmer, Ann. Rev. Nucl. Part. Sci. **40**, 529 (1990).

[7] P. Chen, Phys. Rev. **D46**, 1186 (1992).

[8] Tesla Collaboration, Conceptual Design Report, in preparation.

[9] Desy-Darmstadt Linear Collider Collaboration, Conceptual Design Report, in preparation.

[10] JLC Group, KEK Report 92-16.

[11] NLC ZDR Design Group, SLAC-Report-474.

[12] NLC ZDR Design Group and NLC Physics Working Groups, SLAC-Report-485.

[13] Particle Data Group, Phys. Rev. **D50**, 1173 (1994).

[14] G. Altarelli, R. Kleiss, and C. Verzegnassi, CERN Yellow Report 89-08.

[15] F. James and M. Roos, *MINUIT, Function Minimization and Error Analysis, Release 89.12j*, CERN, Geneva, 1989.

[16] H. Anlauf, IKDA 96/6.

[17] H. Anlauf, private communication.

[18] A. Atkinson and J. Whittaker, Appl. Stat. **28**, 90 (1979).

[19] D. E. Knuth, *Literate Programming*, Vol. 27 of *CSLI Lecture Notes* (Center for the Study of Language and Information, Leland Stanford Junior University, Stanford, CA, 1991).

[20] D. E. Knuth, *TEX: The Program*, Vol. B of *Computers & Typesetting* (Addison-Wesley, Reading, Mass., 1986).

[21] D. E. Knuth, *METAFONT: The Program*, Vol. D of *Computers & Typesetting* (Addison-Wesley, Reading, Mass., 1986).

[22] N. Ramsey, IEEE Software **11**, 97 (1994).

A Literate Programming

A.1 Paradigm

I have presented the sample code in this paper using the *literate programming* paradigm. This paradigm has been introduced by Donald Knuth [19] and his programs TEX [20] and METAFONT [21] provide excellent examples of the virtues of literate programming. Knuth summarized his intention as follows ([19], p. 99)

> "Let us change our traditional attitude to the construction of programs. Instead of imagining that our main task is to instruct a *computer* what to do, let us concentrate rather on explaining to *human beings* what we want a computer to do."

Usually, literate programming uses two utility programs to produce two kinds of files from the source

`tangle` produces the computer program that is acceptable to an "illiterate" (Fortran, C, etc.) compiler. This process consists of stripping documentation and reordering code. Therefore it frees the author from having to present the code in the particular order enforced by a compiler for purely technical reasons. Instead, the author can present the code in the order that is most comprehensible.

`weave` produces a documents that describes the program. Extensive cross referencing of the code sections is usually provided, which has been suppressed in this paper. If a powerful typesetting system (such a TEX) is used, the document can present the algorithms in clear mathematical notation alongside the code. These features improve readability and maintainability of scientific code immensely.

A.2 Practice

Klρχη uses the `noweb` [22] system. This system has the advantage to work with any traditional programming language and support the essential features described in section A.1 with minimal effort. `noweb`'s `tangle` program only reorders the code sections, but does not reformat them. Therefore its output can be used just like any other "illiterate" program.

The examples above should be almost self-explaining, but in order to avoid any ambiguities, I give another example:

37a ⟨*Literate programming example* 37a⟩≡
⟨*Code that has to be at the top* 37c⟩
⟨*Other code* 37b⟩

I can start the presentation with the first line of the "other code":

37b ⟨*Other code* 37b⟩≡ (37a) 37d▷
```
line 1 of the other code
```

If appropriate, the first line of the code that has to appear *before* the other code can be presented later:

37c ⟨*Code that has to be at the top* 37c⟩≡ (37a) 37e▷
```
line 1 of the code at the top
```

Now I can augment the sections:

37d ⟨*Other code* 37b⟩+≡ (37a) ◁37b
```
line 2 of the other code
```

37e ⟨*Code that has to be at the top* 37c⟩+≡ (37a) ◁37c
```
line 2 of the code at the top
```

The complete "program" will be presented to the compiler as

```
line 1 of the code at the top
line 2 of the code at the top
line 1 of the other code
line 2 of the other code
```

The examples in section 3.1.1 show that this reordering is particularly useful for declaring variables when they are first used (rather than at the beginning) and for zooming in on code inside of loops.

B Fortran Name Space

In addition to the ten procedures and one `common` block discussed in section 3

- `circe`, `circee`, `circeg`, `circgg`,
- `girce`, `gircee`, `girceg`, `gircgg`,
- `circes`, `circel`, `/circom/`,

there are two more globally visible functions which are used internally:

- `circem`: error message handler,
- `girceb`: efficient Beta distribution generator.

Even if the `/circom/` is globally visible, application programs *must not* manipulate it directly. The `circes`, subroutine is provided for this purpose and updates some internal parameters as well.

With features from the current Fortran standard (Fortran90), I could have kept the last two functions and the `common` block private. But since Fortran90 has only been adopted by a small fraction of the high energy physics community, I have decided to remain in the confines of Fortran77 (except for the ubiquitous `implicit none`).

Application programs wishing to remain compatible with future versions of Κίρκη must not use `common` blocks or procedures starting with `circe` or `girce`.

C Updates

Information about updates can be obtained

- on the World Wide Web:

  `http://crunch.ikp.physik.th-darmstadt.de/nlc/beam.html`

- by internet FTP:

  host: `crunch.ikp.physik.th-darmstadt.de`

  user: `anonymous`

  password: your email address

  directory: `pub/ohl/circe`

- from mailing lists:

  `circe-announce@crunch.ikp.physik.th-darmstadt.de`

  `circe-bugs@crunch.ikp.physik.th-darmstadt.de`

  `circe-discuss@crunch.ikp.physik.th-darmstadt.de`

  Subscriptions are available from

  `majordomo@crunch.ikp.physik.th-darmstadt.de`

Contributions of results from other simulation programs and updated accelerator designs are welcome at

`Thorsten.Ohl@Physik.TH-Darmstadt.de`