pandora:

do-it-yourself linear collider event generation

M. E. Peskin
May, 2003

In this lecture, I will describe the event generator pandora, which simulates processes at e^+e^-, e^-e^-, $e\gamma$, and $\gamma\gamma$ colliders.

pandora is constructed with the following goals:

- it contains a parametrization of beam effects interfaced to physics processes in a simple way.

- it correctly represents polarization and final state spin effects.

- written in C++, its code is (hopefully) readable and even extensible.

One item is not on the list: precision calculation. pandora achieves only tree-level accuracy.

For any process, a cross section is a convolution of beam distributions with a scattering cross section.

$$\sigma = \int dx_i \; dy_j \; dz_k \; \frac{df^a}{dx_i} \; \frac{df^b}{dy_j} \; \frac{d\sigma^{ab}}{dz_k}$$

pandora assigns the components to C++ classes:

beam distributions	**`beam`**
cross sections	**`process`**

The **`pandora`** class is a Monte Carlo integrator, based on VEGAS.
Its constructor is

```
pandora(beam & B1, beam & B2, process & Pr);
```

Its most important methods are:

```
LEvent getEvent();
LEvent getEvent(double & weight);
double integral(double & sd);
```

```
#include "pandora.h"
#include "eetottbar.h"

int main(){
 double ECM = 500.0;
 double epol = -0.8;  double ppol = 0.0;
 ebeam B1(ECM/2.0, epol,electron, electron);
 B1.setup(NLC500H);
 ebeam B2(ECM/2.0, ppol,positron, positron);
 B2.setup(NLC500H);
 eetottbar Pr;
 pandora P(B1,B2,Pr);
 P.prepare(100000);
 for(int i = 1; i <= 10; i++){
   LEvent E = P.getEvent();   cout << E;
 }
 return 0;
}
```

The LEvent is constructed so that pandora's parton-level events can be straightforwardly input to PYTHIA for fragmentation.

An interface pandora-pythia, written by Masako Iwasaki

inserts the pandora process into PYTHIA as a subprocess.

makes the indicated color string connections

calls out the indicated QCD showers

sends τ s to TAUOLA for decay w. longitudinal polarization.

PYTHIA hadronizes the events and outputs them in StdHep format.

```
#include "pandora.h"
#include "ebeams.h"
#include "eetottbar.h"
#include "pandorarun.h"

int main(int argc, char * argv[]){
 int nEvent   =  atoi(argv[1]);
 char * outfile = argv[2];
 int  iseed_pan = atoi(argv[3]);
/* insert pandora code to define pandora P */
 pandorarun PR(P,nEvent,iseed_pan);
 PR.initialize();
 PR.getevents();
 PR.terminate();
 return 0;
}
```

pandora includes e-, e+ beams with

beamstrahlung - see below

bremsstrahlung - approx. of collinear radiation,
using Fadin-Kuraev structure function

energy spread - approx. of flat-top distribution

default beams include all three features, but these can be turned off individually in simulations:

e.g. B1.ISRoff();

pandora now offers beamstrahlung in three versions:

Yokoya-Chen approximate formulae

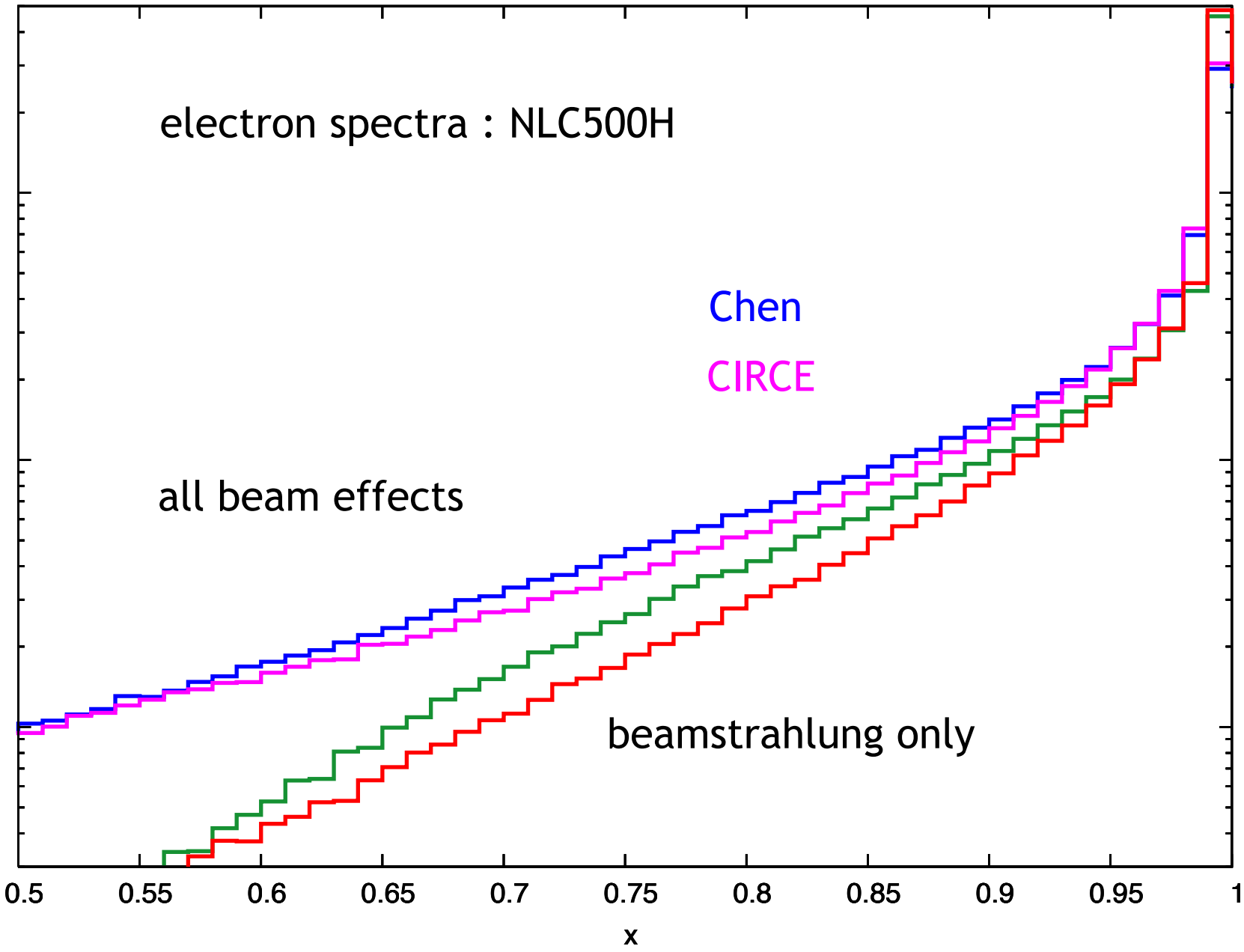
Ohl's CIRCE 1.0 parametrization of Schulte's Guinea-Pig simulations for specific machine designs

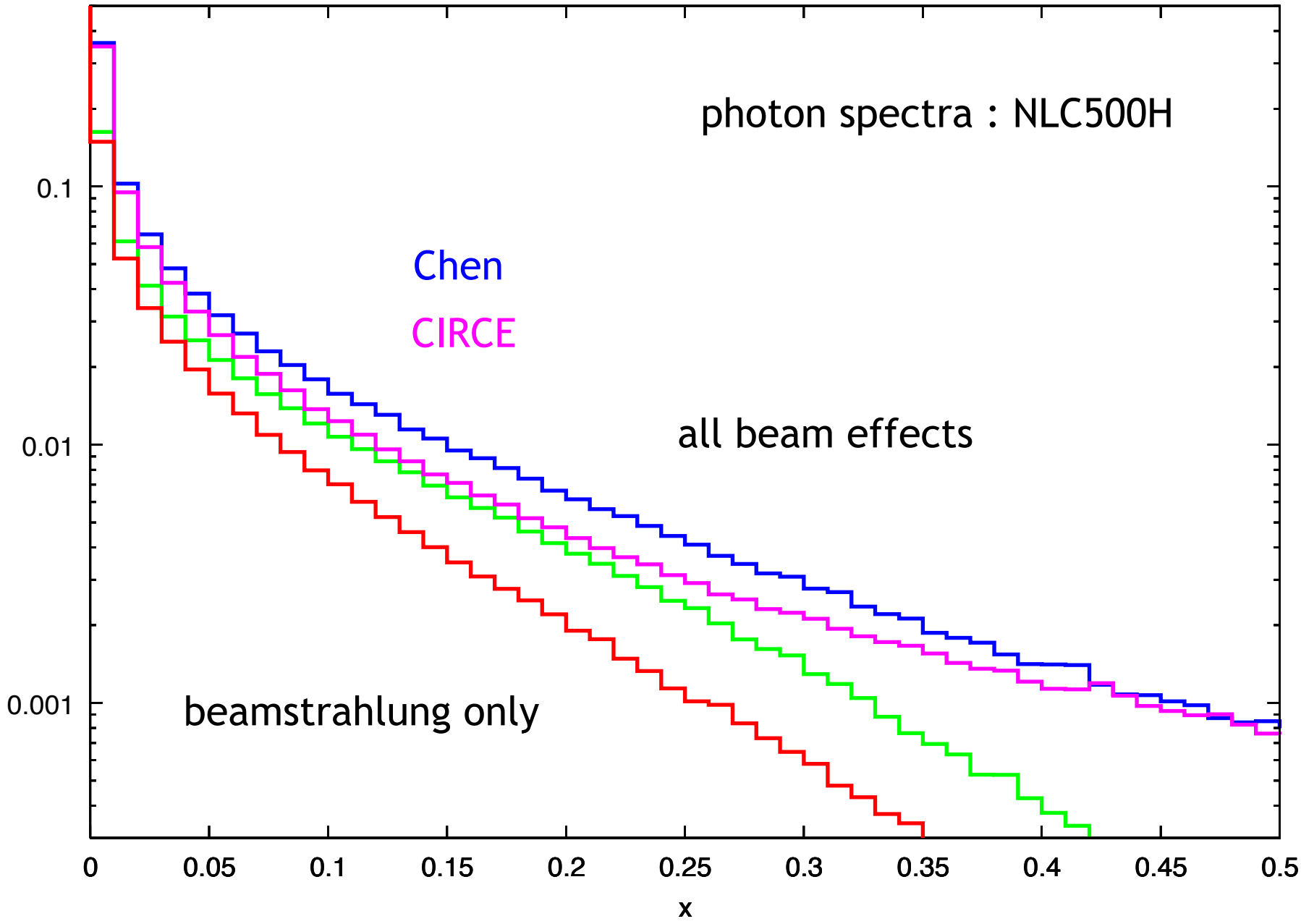
user-defined beams with virtual functions `myF(x)`, `myG(x)`

Ohl's are the best fit to the simulation data.

Yokoya-Chen allows continuous variation of accelerator parameters for studies of beam effects.

pandora also allows input of distributions with beam-beam correlations (luminosity class). Gronberg has built such beam classes from CAIN data for $\gamma\,\gamma$ studies.

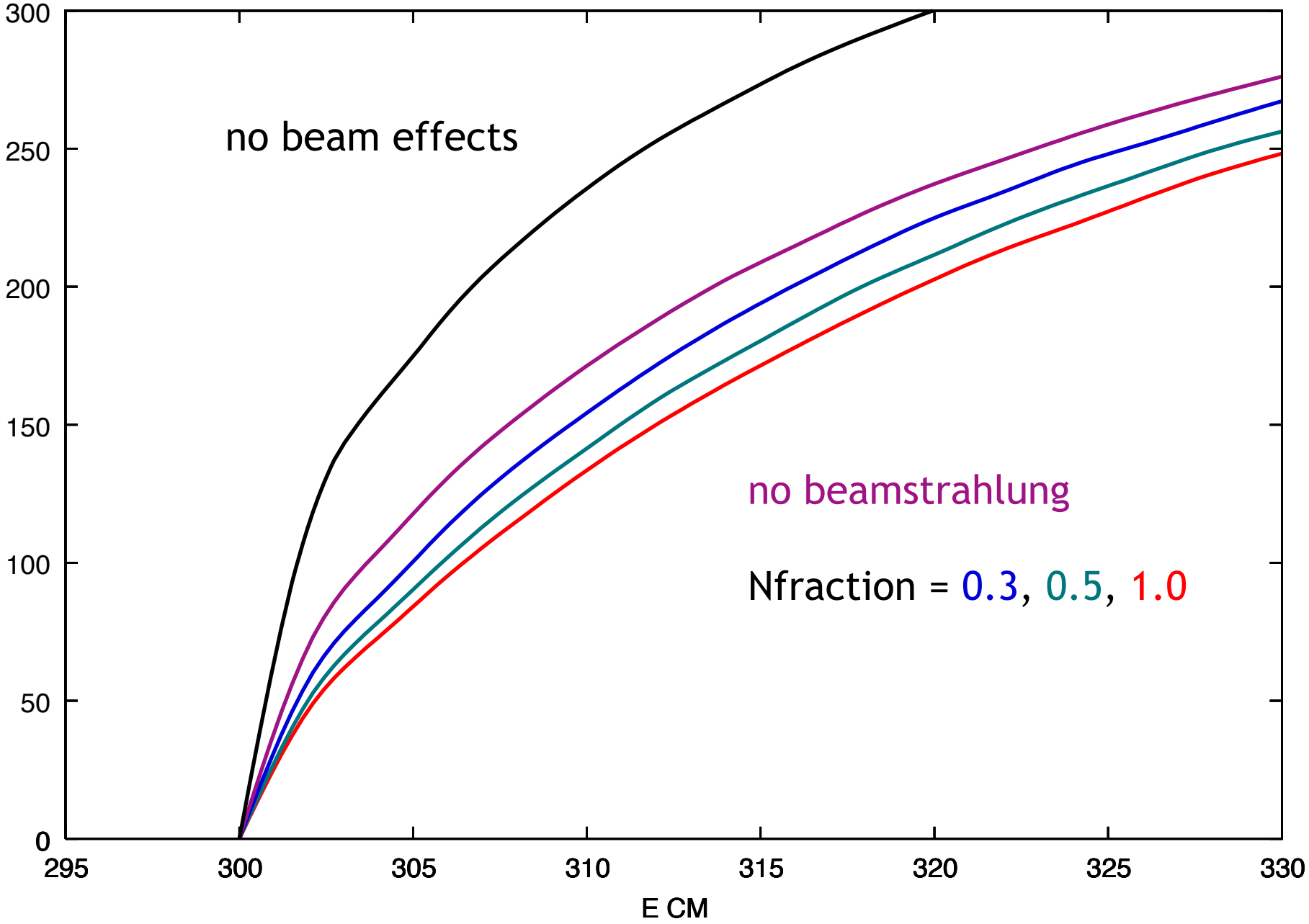

electron spectra : NLC500H
Chen
CIRCE
all beam effects
beamstrahlung only
0.5
0.55
0.6
0.65
0.7
0.75
0.8
0.85
0.9
0.95
1
x


photon spectra : NLC500H
Chen
CIRCE
all beam effects
beamstrahlung only
0.1
0.01
0.001
0
0.05
0.1
0.15
0.2
0.25
0.3
0.35
0.4
0.45
0.5
x

e^- beams get their information about the accelerator from the setup method. The following forms of this method are available for a Yokoya-Chen beam (ebeam class):

```
B.setup(design);                    e.g.   design = NLC500H

B.setup(design, lumi);              returns  luminosity

B.setup(design,Nfraction,lumi);

B.setup(Upsilon,Ngamma,spread);

B.setup(N,sigmax,sigmay,sigmaz, betax,betay,spread,f,lumi);
```

$$e^- e^- \longrightarrow \tilde{e}^- \tilde{e}^-$$



pandora contains a list of predefined standard linear collider physics processes, to be given later.

However, pandora also contains tools to create your own physics processes, or to modify existing ones.

The methods for building new processes are based on C++ class inheritance.

For example,
pandora defines a **fulltdecay** which has Standard Model decays to b_L but allows matrix elements for decay to b_R.

From this, one can easily build a class for non-standard top decay:

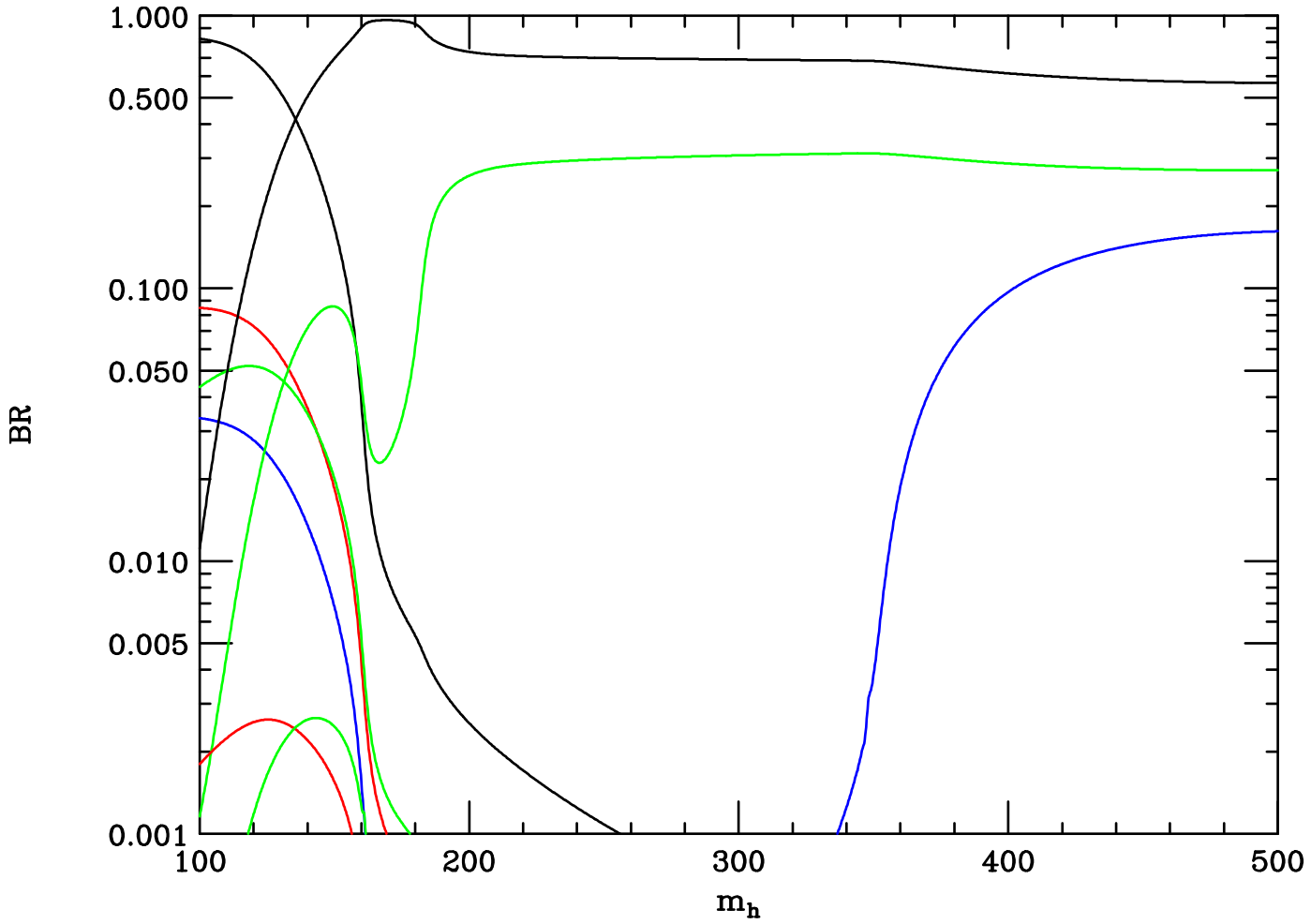
```
tdecayNS : public fulltdecay { ... }

   //   constructor
tdecayNS(FW1L,FW1R,FW2L,FW2R);
  //   overload
void properamplitudes(){
    CDPamps[-1][1] =
        sqrt(2.0 * (1.0 + coschi))
          (DecayFs[1]-0.5*DecayFs[3]);
    ...
}
```

For particles with several different decay modes, pandora includes a class **complexdecay**. This includes an array of decay classes representing specific decay modes and a **noodle** which chooses one mode for each particular event. Some methods of this class are:

```
void addChannel(decay * D);

double partialGamma(int i);
DVector partialGammas();

void newGammaValue(int i, double GV);
void newGammaPattern(DVector & GVs);

void makestable();
```

Higgsdecay is a subclass of **complexdecay** which includes 10 Standard Model decay channels.

Higgs branching ratio pattern from **Higgsdecay**



Processes implemented in the current version of pandora: pandora 2.3 / pandora-pythia 3.3

$e^+e^- \to \; l^+l^- \;\; qq \;\; e^+e^- \;\; \gamma\gamma \;\; t\bar{t} \;\; \gamma Z^0 \;\; Z^0 Z^0 \;\; W^+W^-$

$\gamma\,\gamma \to \; l^+l^- \;\; qq \;\; e^+e^- \;\; t\bar{t} \;\; W^+W^-$

$e\,\gamma \to \; e\,\gamma \;\; e\,Z^0 \;\; \nu W$

$e^+e^- \to \; Z^0 h^0 \;\; \nu\,\bar{\nu}\,h^0 \;\; e^+e^- h^0$

$e^+e^- \to \; \gamma\,\nu\,\bar{\nu}$

and a few beyond-the-Standard-Model processes, e.g.:

$e^+e^- \to \; l^+l^- \;\; qq \;\; e^+e^-$

with E_6 Z' or graviton exchange

$e^+e^- \to \; \gamma G$

beam classes for e^+e^-, e^-e^-, $\gamma\gamma$.

Find the latest version at pandora's home page:

http://www-sldnt.slac.stanford.edu/
nld/new/Docs/Generators/PANDORA.htm

or use the link from my home page:

http://www.slac.stanford.edu/~mpeskin/

On pandora's home page you will find links to download:

- pandora 2.3
- pandora-pythia 3.3
- the pandora/pandora-pythia user's guide (pdf)