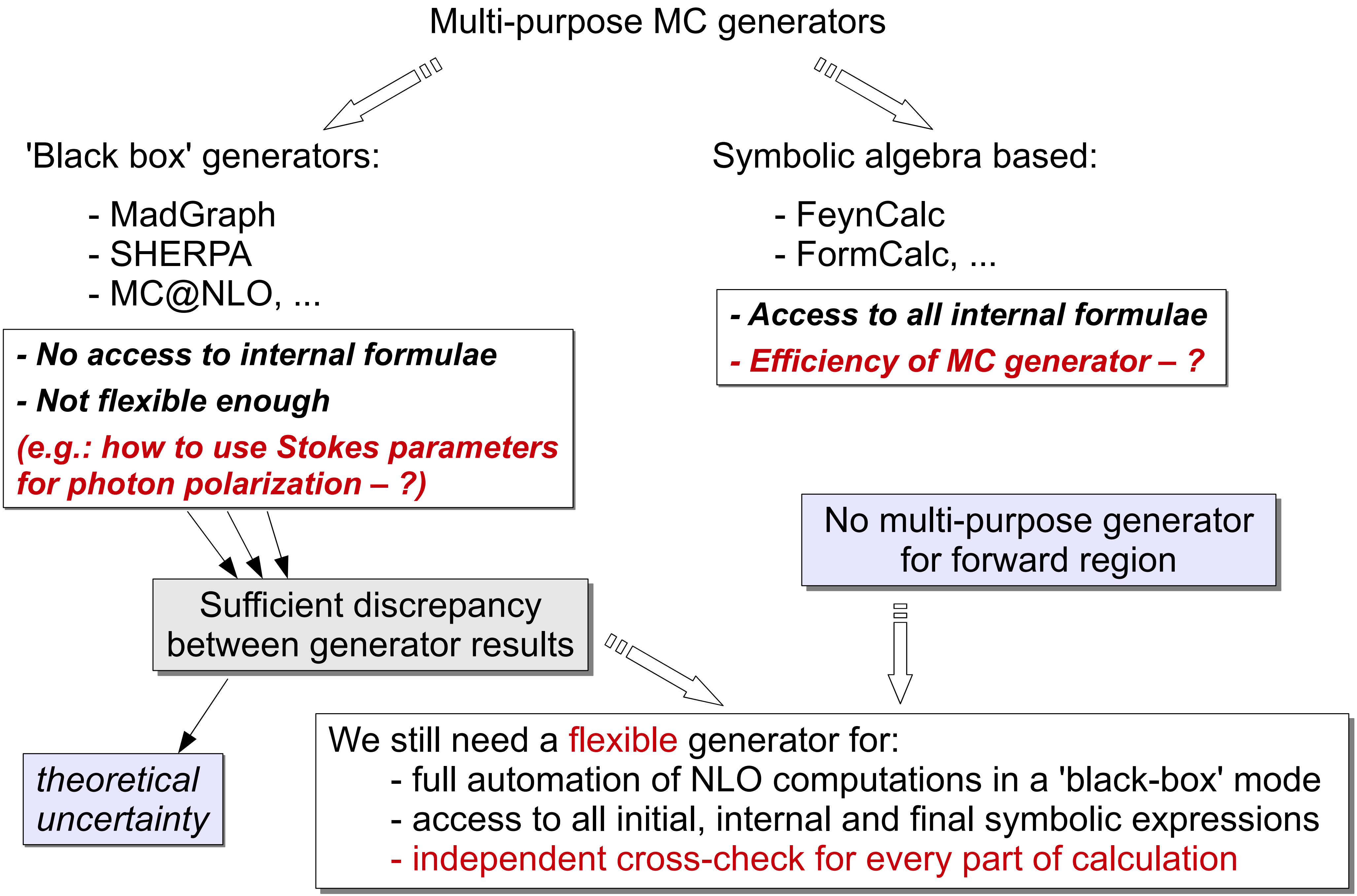
Using the ALHEP program for calculation of one-loop radiative corrections

V. Makarenko

NC PHEP BSU, Minsk

Motivation



Multi-purpose MC generators

'Black box' generators:

- MadGraph
- SHERPA
- MC@NLO, ...

- No access to internal formulae

- Not flexible enough

(e.g.: how to use Stokes parameters for photon polarization – ?)

Symbolic algebra based:

- FeynCalc
- FormCalc, ...

- Access to all internal formulae

- Efficiency of MC generator – ?

Sufficient discrepancy between generator results

theoretical uncertainty

No multi-purpose generator for forward region

We still need a flexible generator for:

- full automation of NLO computations in a 'black-box' mode
- access to all initial, internal and final symbolic expressions
- independent cross-check for every part of calculation

ALHEP algebra core

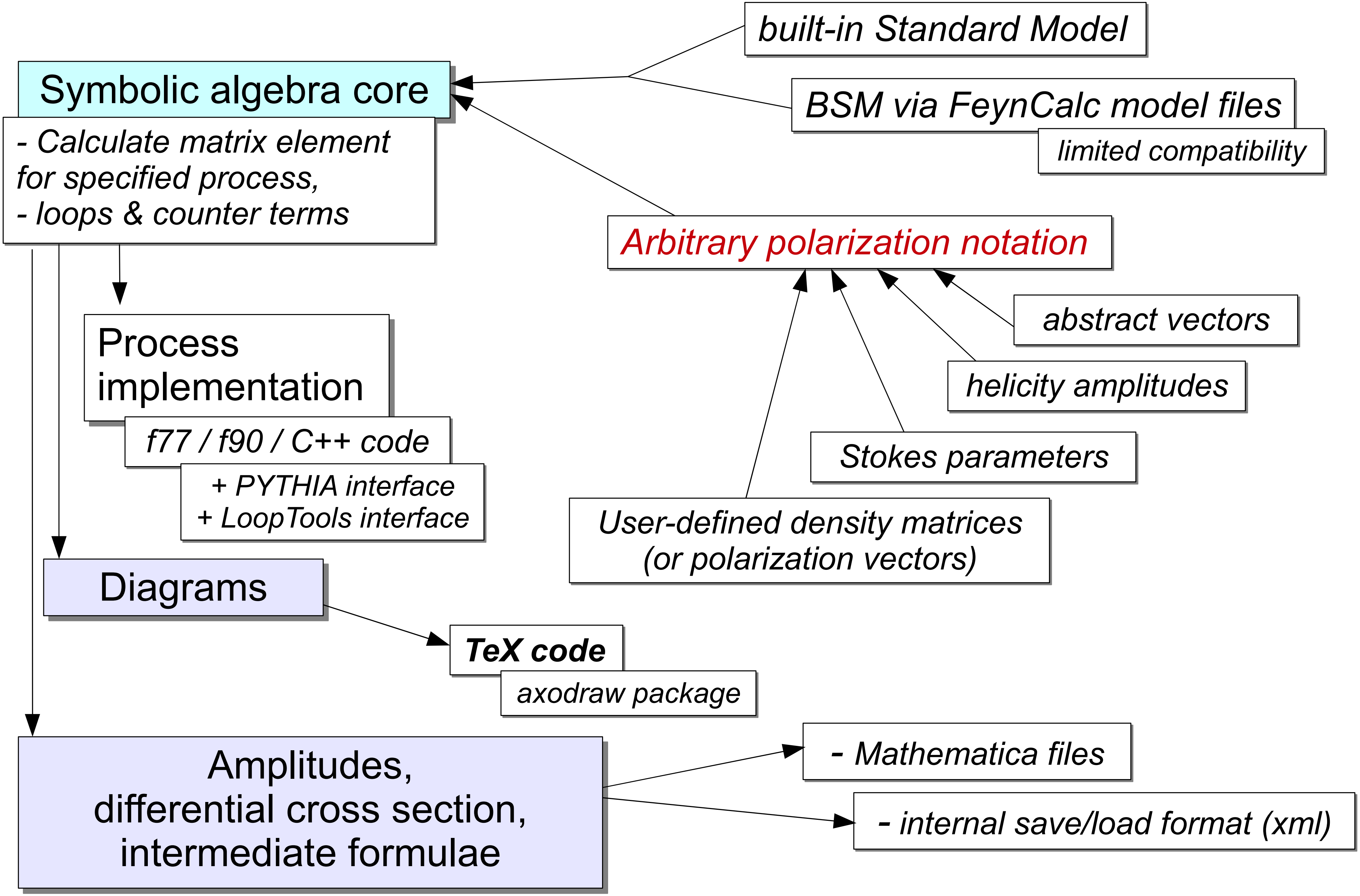
Symbolic algebra core

Pure C++ logic is more flexible than Mathematica, FORM, ...

- *Calculate amplitudes or squared matrix element*
- *evaluate traces*
- *simplify / minimize number of gamma-matrices in expression*
- *N-dimensional evaluation of loop diagrams*
- *reduce tensor virtual integrals to scalar ones*
  - *try to replace numerator with a sum of denominators (using kinematic relations)*
  - *solve linear system for general case*
  - *or retain complicated tensor integrals (and use LoopTools for them)*
- *simplify using kinematic relations between momenta & couplings*
  - *find the shortest representation for expressions*
- *minimize number of sum-and-multiply operations*
  - *for faster numerical code*
- *create Mathematica code for symbolic expressions*
- *create C++ or Fortran code for numerical analysis*

http://www.hep.by/alhep

ALHEP program



Symbolic algebra core

- Calculate matrix element for specified process,
- loops & counter terms

built-in Standard Model

BSM via FeynCalc model files

limited compatibility

Arbitrary polarization notation

abstract vectors

helicity amplitudes

Stokes parameters

User-defined density matrices
(or polarization vectors)

Process implementation

f77 / f90 / C++ code

+ PYTHIA interface
+ LoopTools interface

Diagrams

TeX code

axodraw package

Amplitudes,
differential cross section,
intermediate formulae

- Mathematica files

- internal save/load format (xml)

Scripts & diagrams

C-like command script

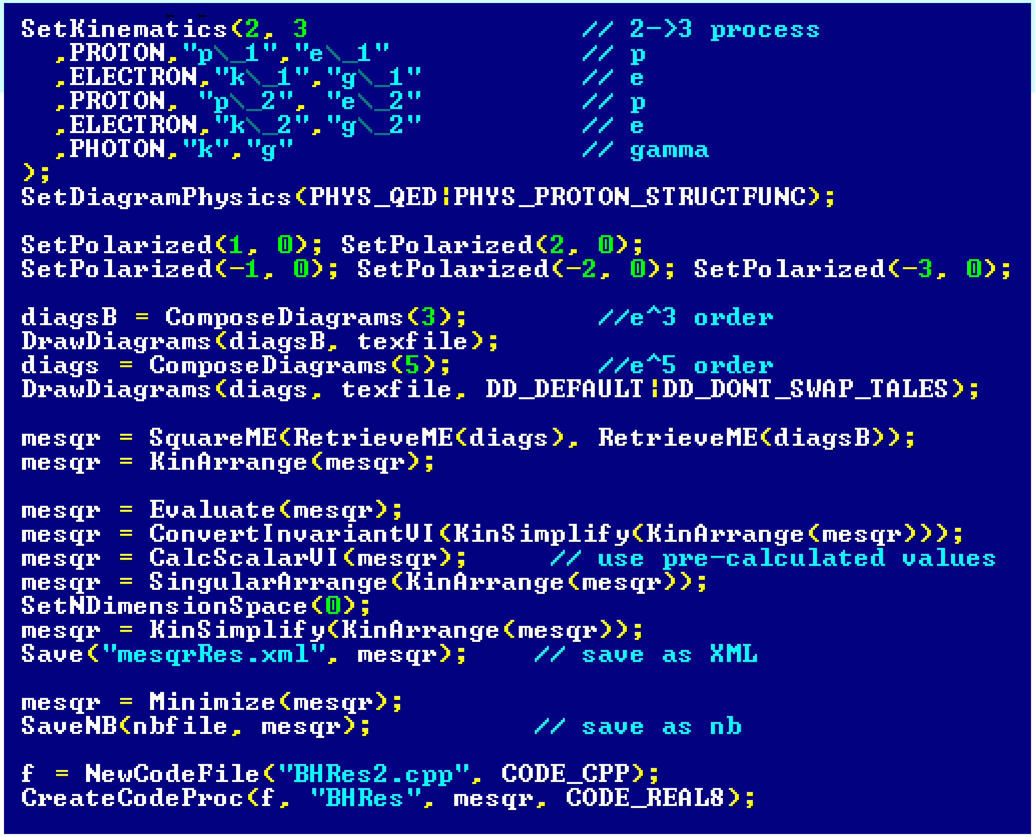
Symbolic algebra core

- Calculate matrix element for specified process,
- loops & counter terms

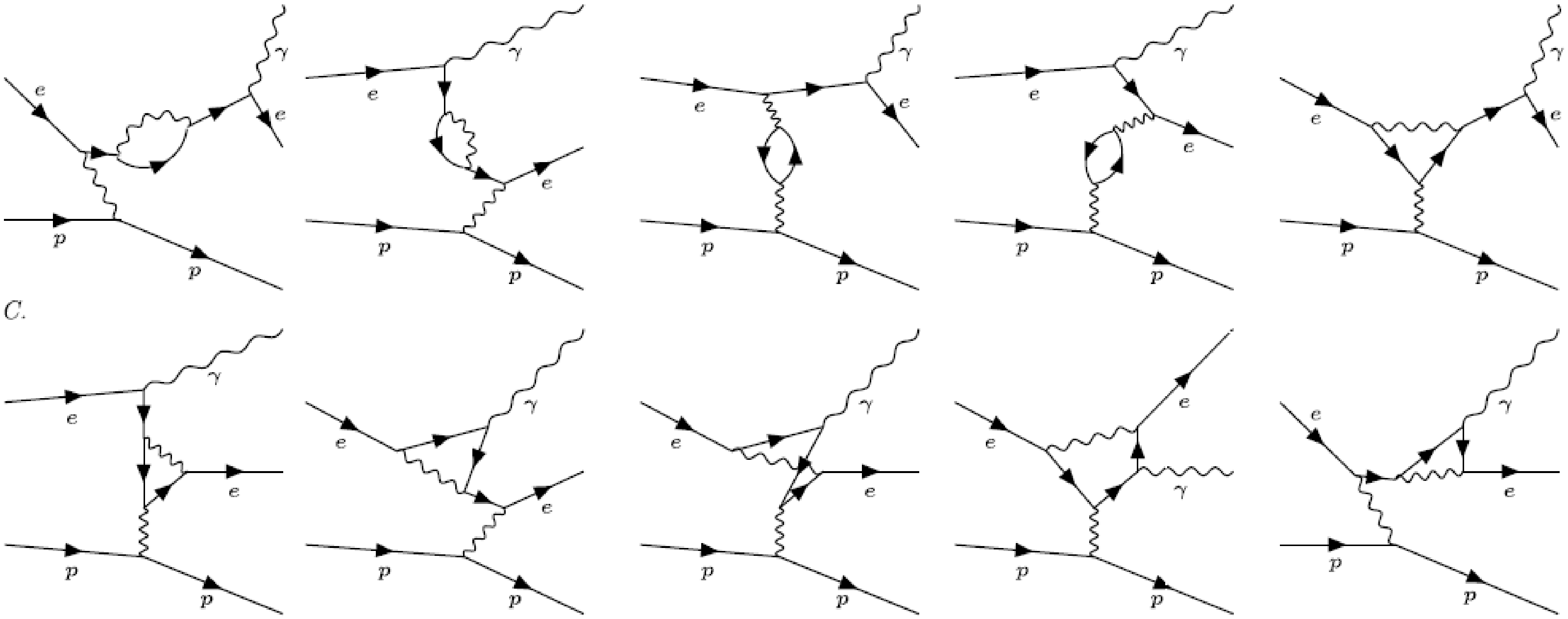
Process implementation

f77 / f90 / C++ code

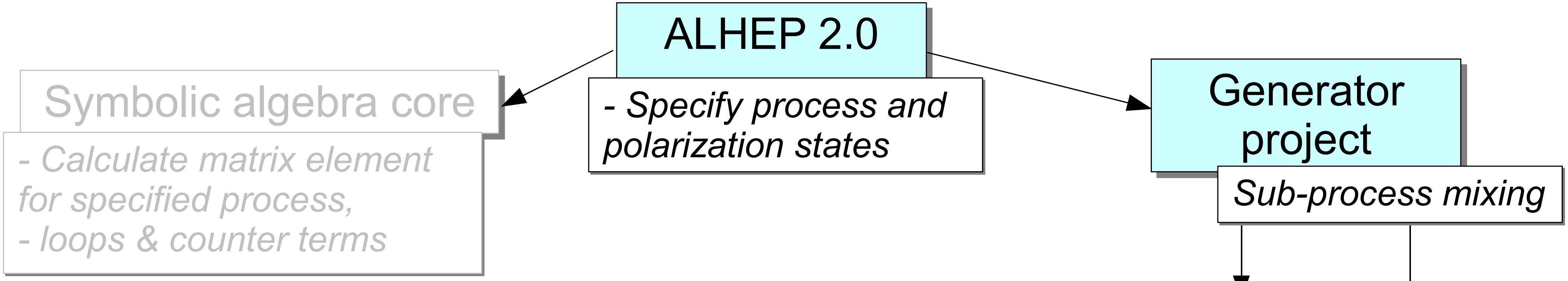
Diagrams
TeX code



```
SetKinematics(2, 3                 // 2->3 process
  ,PROTON,"p\_1","e\_1"            // p
  ,ELECTRON,"k\_1","g\_1"          // e
  ,PROTON, "p\_2", "e\_2"          // p
  ,ELECTRON,"k\_2","g\_2"          // e
  ,PHOTON,"k","g"                  // gamma
);
SetDiagramPhysics(PHYS_QED|PHYS_PROTON_STRUCTFUNC);

SetPolarized(1, 0); SetPolarized(2, 0);
SetPolarized(-1, 0); SetPolarized(-2, 0); SetPolarized(-3, 0);

diagsB = ComposeDiagrams(3);        //e^3 order
DrawDiagrams(diagsB, texfile);
diags  = ComposeDiagrams(5);        //e^5 order
DrawDiagrams(diags, texfile, DD_DEFAULT|DD_DONT_SWAP_TALES);

mesqr  = SquareME(RetrieveME(diags), RetrieveME(diagsB));
mesqr  = KinArrange(mesqr);

mesqr  = Evaluate(mesqr);
mesqr  = ConvertInvariantVI(KinSimplify(KinArrange(mesqr)));
mesqr  = CalcScalarVI(mesqr);    // use pre-calculated values
mesqr  = SingularArrange(KinArrange(mesqr));
SetNDimensionSpace(0);
mesqr  = KinSimplify(KinArrange(mesqr));
Save("mesqrRes.xml", mesqr);    // save as XML

mesqr  = Minimize(mesqr);
SaveNB(nbfile, mesqr);          // save as nb

f  = NewCodeFile("BHRes2.cpp", CODE_CPP);
CreateCodeProc(f, "BHRes", mesqr, CODE_REAL8);
```



ALHEP version 2.0



ALHEP 2.0

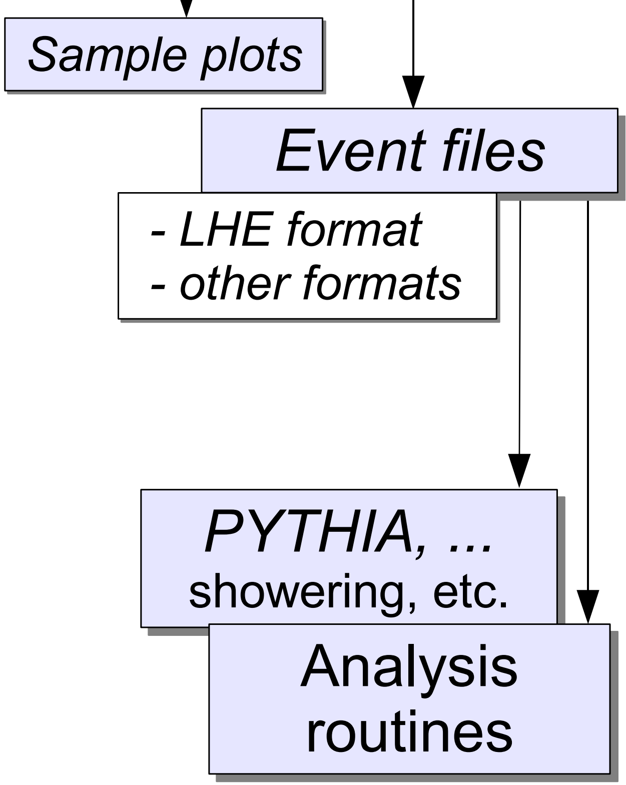
- Specify process and polarization states

Symbolic algebra core

- Calculate matrix element for specified process,
- loops & counter terms

Generator project

Sub-process mixing



Sample plots

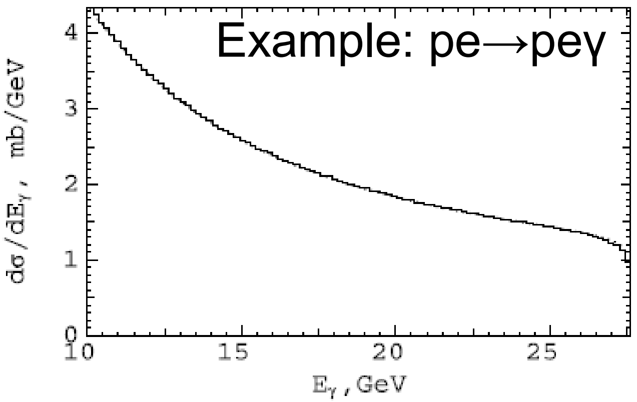
Event files

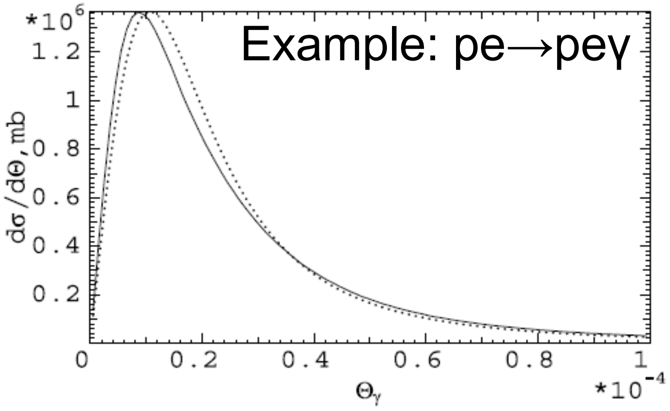
- LHE format
- other formats

PYTHIA, ...
showering, etc.

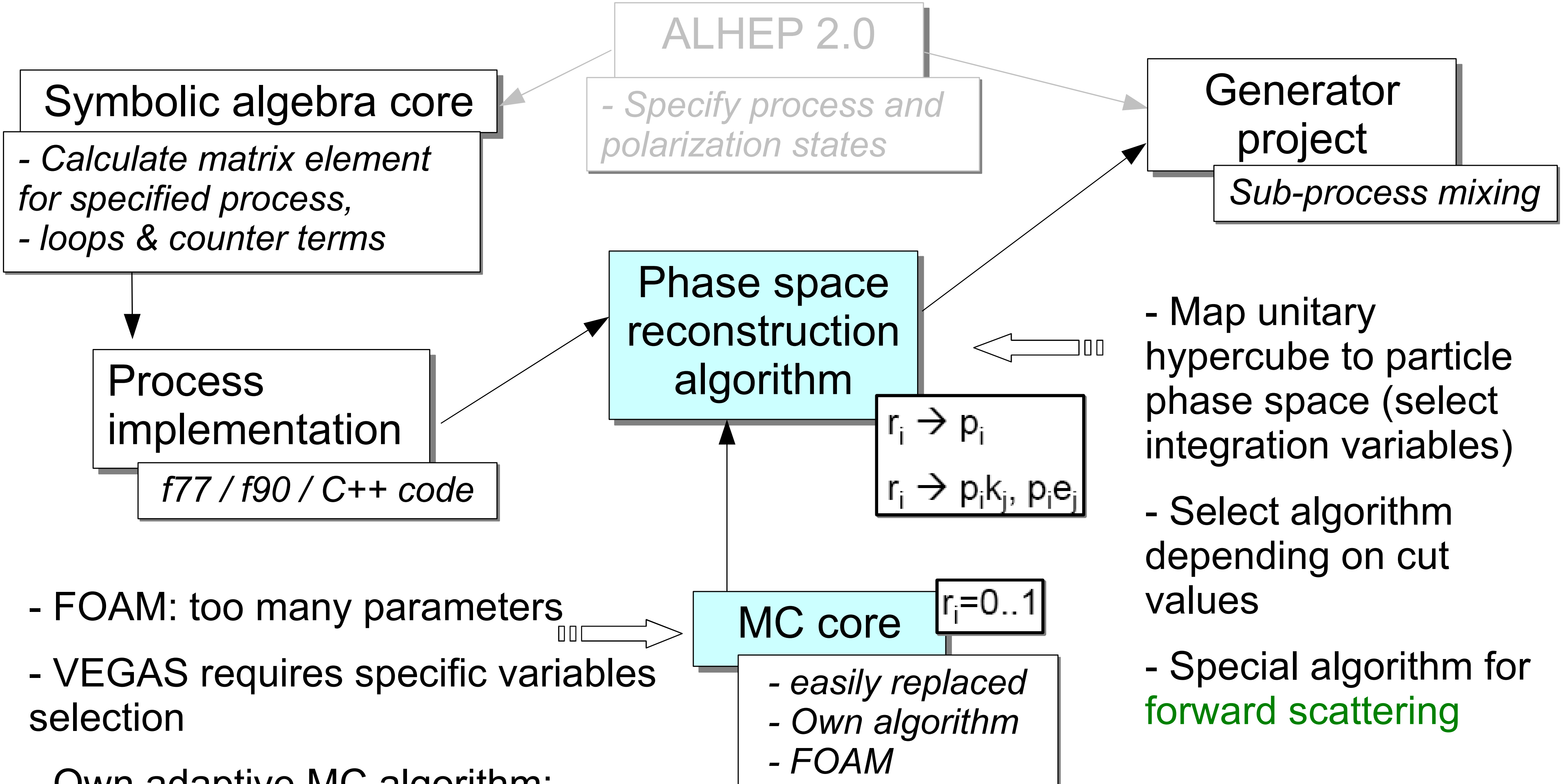
Analysis routines

- Automatic creation of generator project for numerical analysis
- Single 'Run' script command
- Adaptive MC core
- Unweighted event output
- User access to event generation loop code
  + sample histogram filling





Generator structure



ALHEP 2.0

- Specify process and polarization states

Symbolic algebra core

- Calculate matrix element for specified process,
- loops & counter terms

Generator project

Sub-process mixing

Process implementation

f77 / f90 / C++ code

Phase space reconstruction algorithm

$r_i \rightarrow p_i$

$r_i \rightarrow p_i k_j, p_i e_j$

MC core

r_i=0..1

- easily replaced
- Own algorithm
- FOAM
*- VEGAS**

- Map unitary hypercube to particle phase space (select integration variables)

- Select algorithm depending on cut values

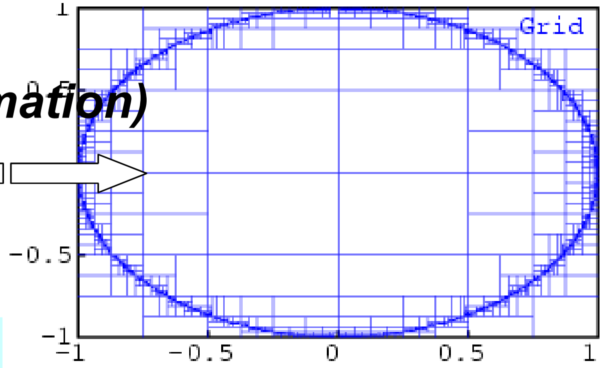
- Special algorithm for forward scattering

- FOAM: too many parameters

- VEGAS requires specific variables selection

- Own adaptive MC algorithm:

Step 1: pre-integration (split volume according to fast approximation)
Step 2: integration (split volume according to exact function)
- uses function derivatives for better peak detection
Step 3: unweighted event generation



Cuts

Applying cuts (as step-functions) in MC:

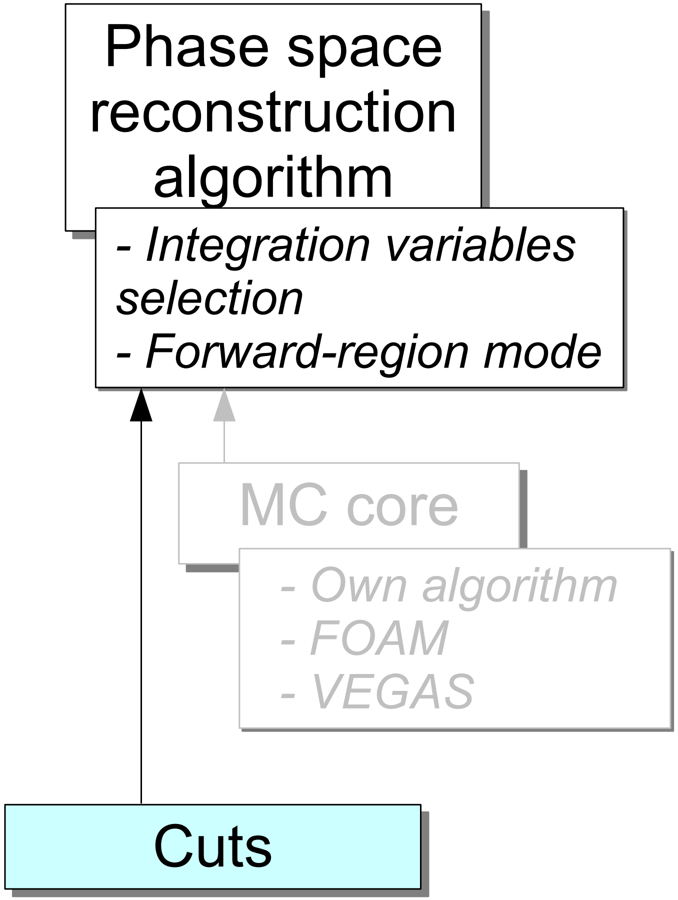
- MC algorithm requires smooth function

- Systematic error may appear in cut-adjacent regions (no gradient value available is for proper cell splitting)

- Cuts applied at the event generation step (only) causes no integration problems, but decreases the efficiency of generator

- Simple cuts may be avoided by smart selection of integration variables

- Some cuts must be applied at integration step


Phase space reconstruction algorithm
- Integration variables selection
- Forward-region mode
MC core
- Own algorithm
- FOAM
- VEGAS
Cuts

Cuts

Applying cuts (as step-functions) in MC:

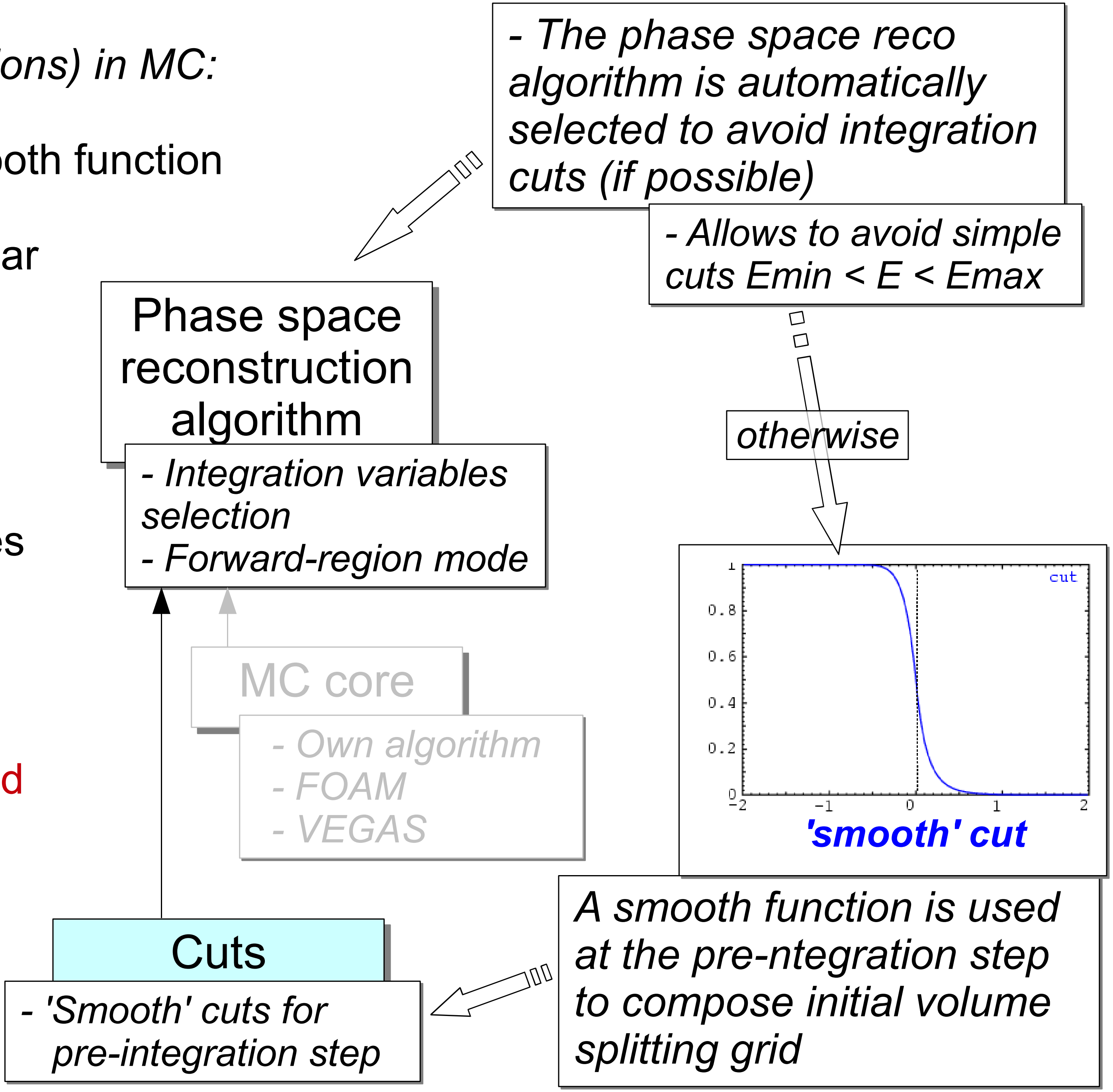
- MC algorithm requires smooth function

- Systematic error may appear in cut-adjacent regions (no gradient value available is for proper cell splitting)

- Cuts applied at the event generation step (only) causes no integration problems, but decreases the efficiency of generator

- Simple cuts may be avoided by smart selection of integration variables

- Some cuts must be applied at integration step



- The phase space reco algorithm is automatically selected to avoid integration cuts (if possible)

- Allows to avoid simple cuts Emin < E < Emax

otherwise

Phase space reconstruction algorithm

- Integration variables selection
- Forward-region mode

MC core

- Own algorithm
- FOAM
- VEGAS

'smooth' cut

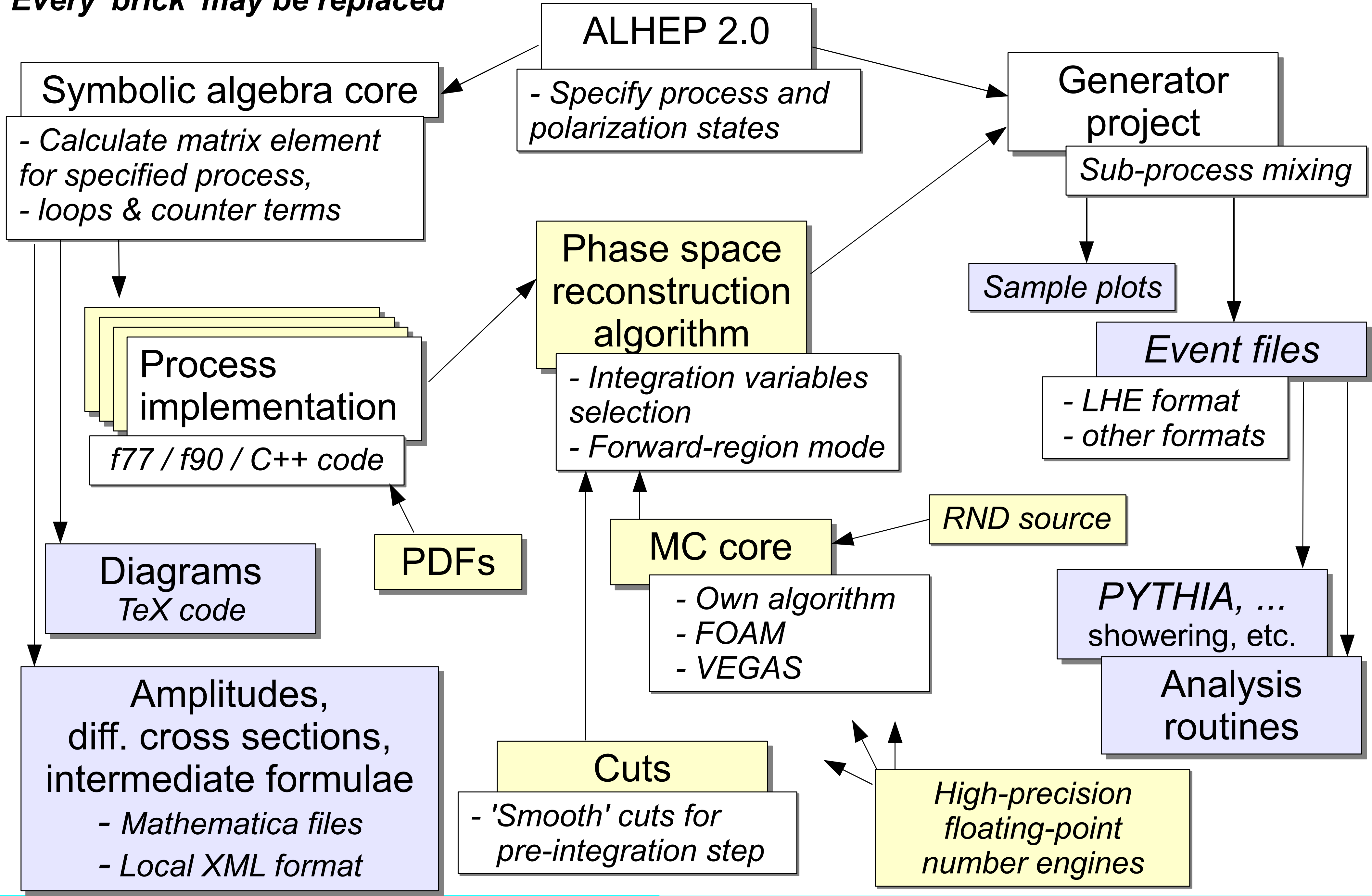
A smooth function is used at the pre-ntegration step to compose initial volume splitting grid

Cuts

- 'Smooth' cuts for pre-integration step

Brick-based architecture

Every 'brick' may be replaced



ALHEP 2.0

- Specify process and polarization states

Symbolic algebra core

- Calculate matrix element for specified process,
- loops & counter terms

Generator project

Sub-process mixing

Phase space reconstruction algorithm

- Integration variables selection
- Forward-region mode

Sample plots

Event files

- LHE format
- other formats

Process implementation

f77 / f90 / C++ code

PDFs

Diagrams
TeX code

MC core

- Own algorithm
- FOAM
- VEGAS

RND source

PYTHIA, ...
showering, etc.

Analysis routines

Amplitudes,
diff. cross sections,
intermediate formulae

- Mathematica files
- Local XML format

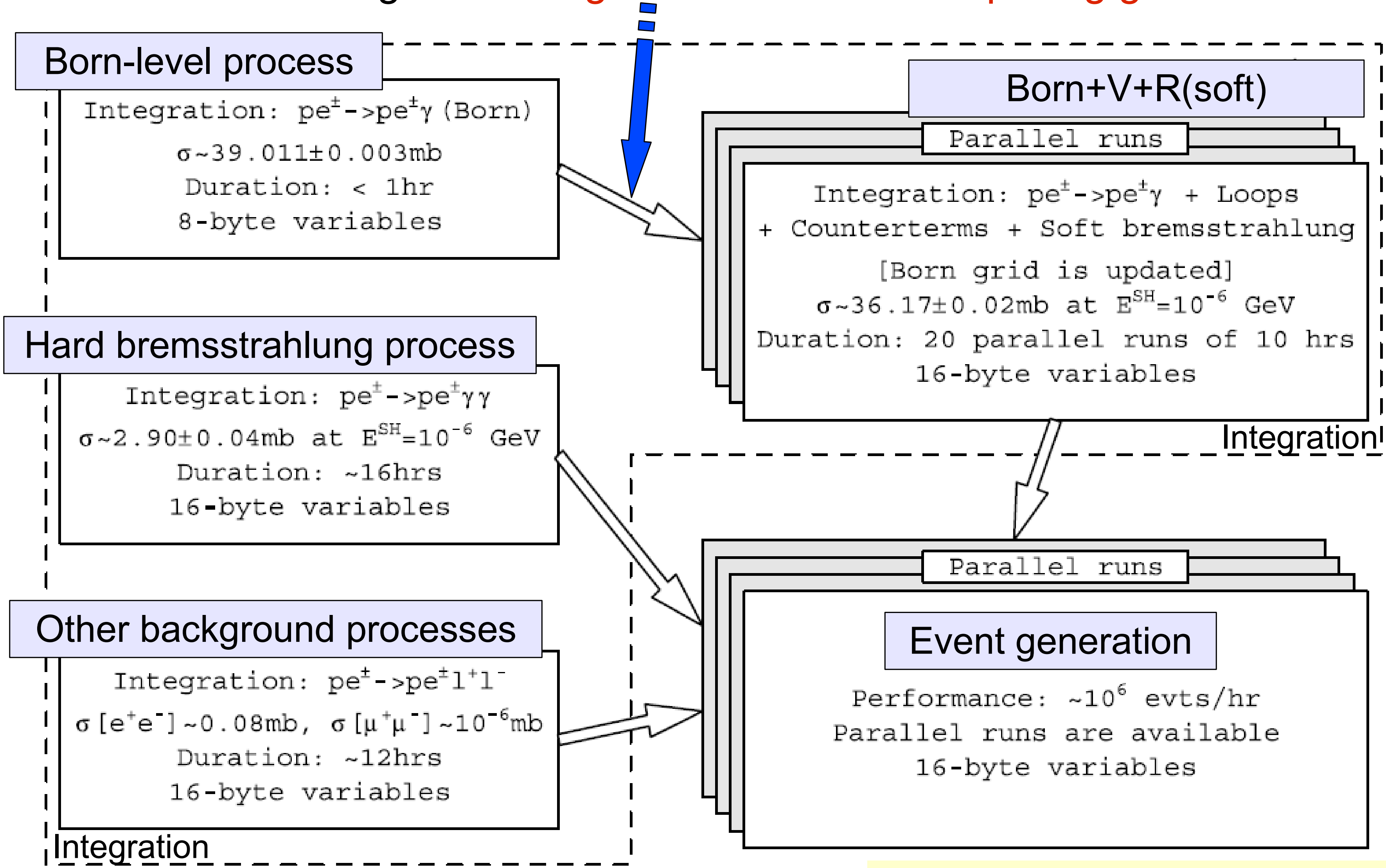
Cuts

- 'Smooth' cuts for pre-integration step

High-precision floating-point number engines

Generator usage example

The differential cross section term Born + Loops + Soft bremsstrahlung is integrated using Born-level volume splitting grid



Born-level process

Integration: $pe^{\pm} \to pe^{\pm}\gamma$ (Born)

$\sigma \sim 39.011 \pm 0.003$mb

Duration: < 1hr

8-byte variables

Hard bremsstrahlung process

Integration: $pe^{\pm} \to pe^{\pm}\gamma\gamma$

$\sigma \sim 2.90 \pm 0.04$mb at $E^{SH}=10^{-6}$ GeV

Duration: ~16hrs

16-byte variables

Other background processes

Integration: $pe^{\pm} \to pe^{\pm}l^{+}l^{-}$

$\sigma[e^{+}e^{-}] \sim 0.08$mb, $\sigma[\mu^{+}\mu^{-}] \sim 10^{-6}$mb

Duration: ~12hrs

16-byte variables

Integration

Born+V+R(soft)

Parallel runs

Integration: $pe^{\pm} \to pe^{\pm}\gamma$ + Loops + Counterterms + Soft bremsstrahlung

[Born grid is updated]

$\sigma \sim 36.17 \pm 0.02$mb at $E^{SH}=10^{-6}$ GeV

Duration: 20 parallel runs of 10 hrs

16-byte variables

Integration

Parallel runs

Event generation

Performance: ~10^{6} evts/hr

Parallel runs are available

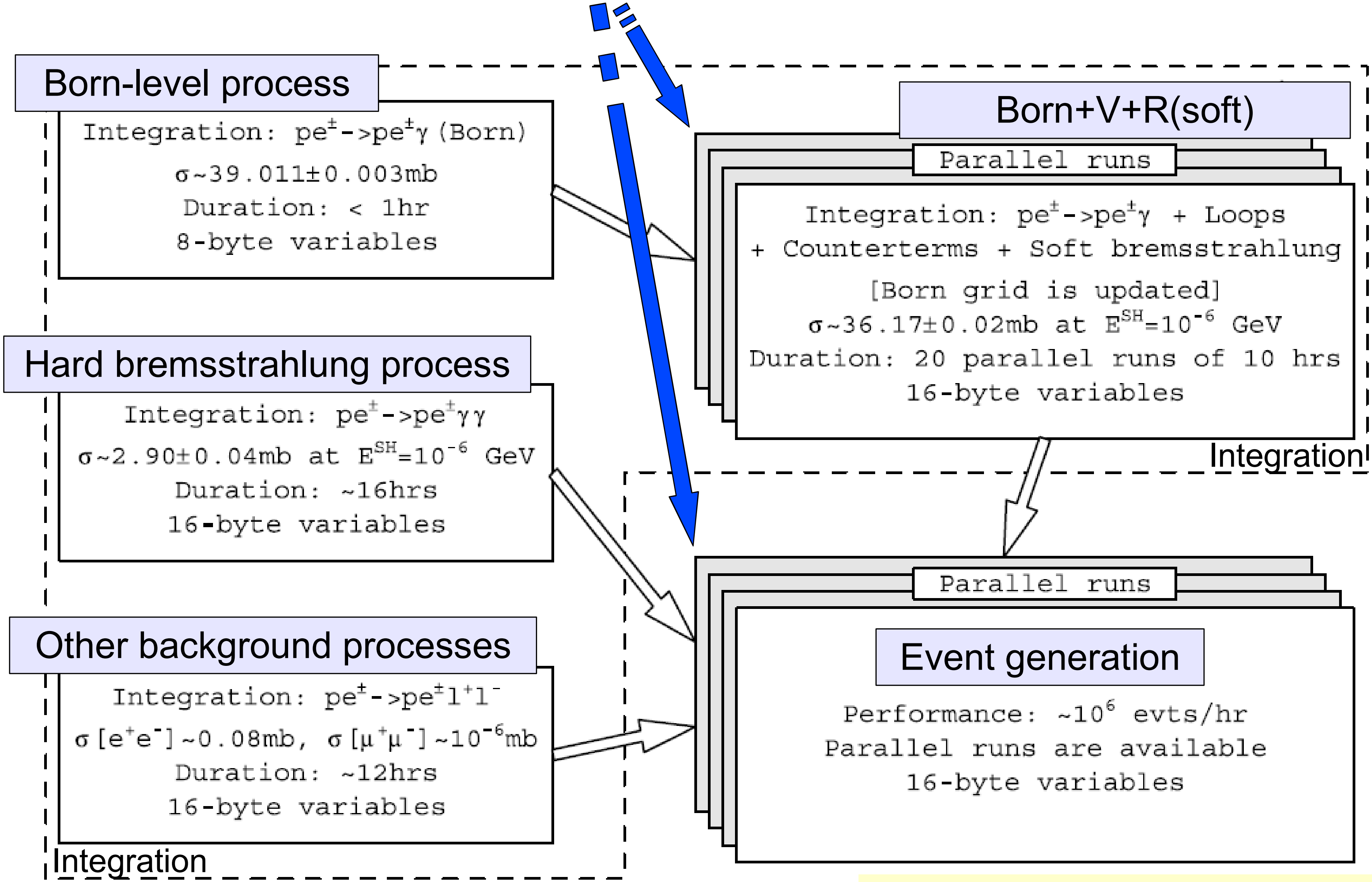
16-byte variables

Example: pe→peγ with forward photon

Eur.Phys.J.C71:1574,2011

Parallel computations

Parallel computations are implemented wherever it is possible



Born-level process

Integration: $pe^{\pm}\to pe^{\pm}\gamma$ (Born)

$\sigma\sim39.011\pm0.003$mb

Duration: < 1hr

8-byte variables

Hard bremsstrahlung process

Integration: $pe^{\pm}\to pe^{\pm}\gamma\gamma$

$\sigma\sim2.90\pm0.04$mb at $E^{SH}=10^{-6}$ GeV

Duration: ~16hrs

16-byte variables

Other background processes

Integration: $pe^{\pm}\to pe^{\pm}l^{+}l^{-}$

$\sigma[e^{+}e^{-}]\sim0.08$mb, $\sigma[\mu^{+}\mu^{-}]\sim10^{-6}$mb

Duration: ~12hrs

16-byte variables

Integration

Born+V+R(soft)

Parallel runs

Integration: $pe^{\pm}\to pe^{\pm}\gamma$ + Loops + Counterterms + Soft bremsstrahlung

[Born grid is updated]

$\sigma\sim36.17\pm0.02$mb at $E^{SH}=10^{-6}$ GeV

Duration: 20 parallel runs of 10 hrs

16-byte variables

Integration

Parallel runs

Event generation

Performance: $\sim10^{6}$ evts/hr

Parallel runs are available

16-byte variables

Example: pe→peγ with forward photon

Eur.Phys.J.C71:1574,2011

Conclusions ...

Basic concepts:

- brick-based generator architecture
  - *allows independent check of every 'brick'*
    - *different ME forms, MC engines etc.*
  - *forward region generator option*
  - *start MC integration with approximate phase space grid*
    - *faster V-term integration*
    - *'smooth' cut option*
- flexible algebra core
  - *arbitrary polarization notation*
  - *access to intermediate symbolic expressions*

Conclusions and current status

Basic concepts:

- brick-based generator architecture
  - *allows independent check of every 'brick'*
    - *different ME forms, MC engines etc.*
  - *forward region generator option*
  - *start MC integration with approximate phase space grid*
    - *faster V-term integration*
    - *'smooth' cut option*
- flexible algebra core
  - *arbitrary polarization notation*
  - *access to intermediate symbolic expressions*

Current status:

- algebraic calculations

- LO – ready

- one-loop RC (including regularization, renormalization, and hard bremsstrahlung) – ready

- BSM processes – more tests are required

- automation of generator composing

- LO – ready

- one-loop RC – partial automation

(manual edition of C++ generator code is still required for some operations: link processes, select separator values, check the generator consistency, parallel runs, …)

- internal MC core – ready

+more debug is required