LUND UNIVERSITY

Workshop: Physics at TeV Colliders
Les Houches
17 May 2005

The Les Houches Accord
Should/Could We Update It?

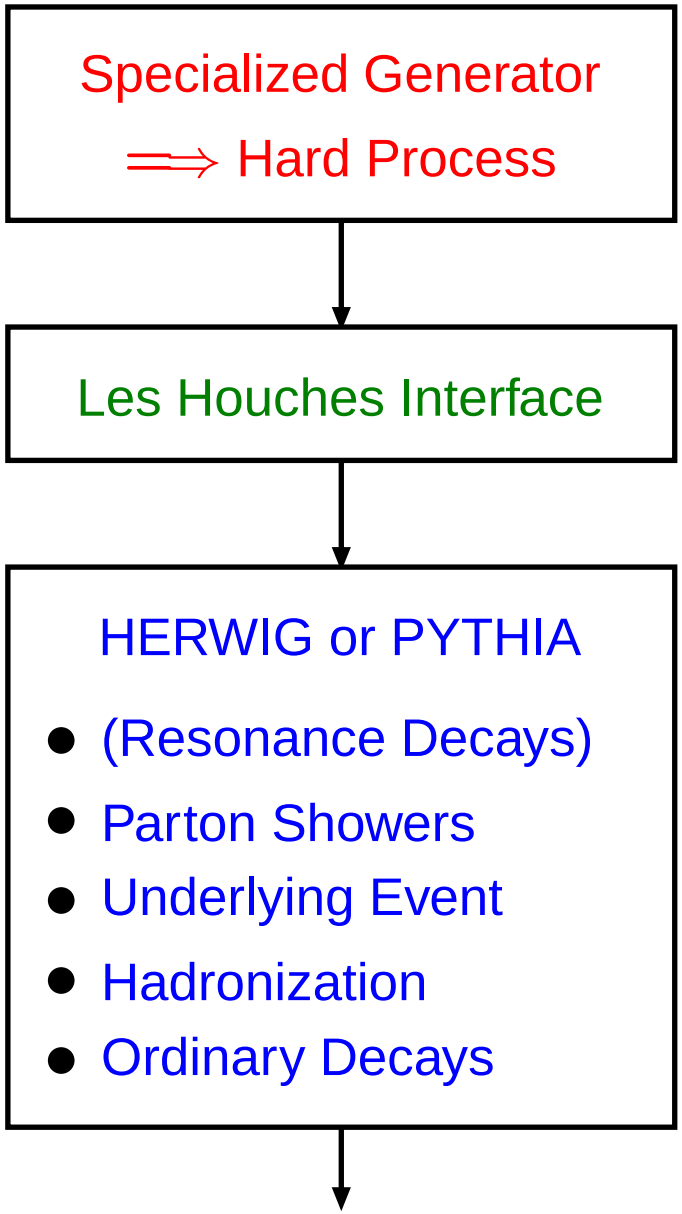
- **Translate from Fortran to C++?**
- Provide further information?
- Standardize file formats?

Torbjörn Sjöstrand

CERN/PH and

Department of Theoretical Physics, Lund University

The Context



Some Specialized Generators:

- AcerMC: $\mathrm{t\bar{t}b\bar{b}}$, ...
- ALPGEN: $\mathrm{W/Z+} \leq 6\mathrm{j}$, $n\mathrm{W} + m\mathrm{Z} + k\mathrm{H+} \leq 3\mathrm{j}$, ...
- AMEGIC++: generic LO
- CompHEP: generic LO
- GRACE+Bases/Spring: generic LO+ some NLO loops
- GR@PPA: $\mathrm{b\bar{b}b\bar{b}}$
- MadCUP: $\mathrm{W/Z+} \leq 3\mathrm{j}, \mathrm{t\bar{t}b\bar{b}}$
- MadGraph+HELAS: generic LO
- MCFM: NLO $\mathrm{W/Z+} \leq 2\mathrm{j}$, WZ, WH, $\mathrm{H+} \leq 1\mathrm{j}$
- O'Mega+WHIZARD: generic LO
- VECBOS: $\mathrm{W/Z+} \leq 4\mathrm{j}$

Apologies for all unlisted programs

The Les Houches Accord

Les Houches accord May 2001 $\Rightarrow$ E Boos et al., hep-ph/0109068

The LHA introduces two steps in a run, where a user can intervene:

1) at initialization, the generator does a

`CALL UPINIT`

where the user will define the character of a run by setting info in

`COMMON/HEPRUP/`

2) for each new event, the generator does a

`CALL UPEVNT`

where the user will define the next event by setting info in

`COMMON/HEPEUP/`

Initialization

```
 INTEGER MAXPUP
 PARAMETER (MAXPUP=100)
 INTEGER IDBMUP,PDFGUP,PDFSUP,IDWTUP,NPRUP,LPRUP
 DOUBLE PRECISION EBMUP,XSECUP,XERRUP,XMAXUP
 COMMON/HEPRUP/IDBMUP(2),EBMUP(2),PDFGUP(2),PDFSUP(2),IDWTUP,
&NPRUP,XSECUP(MAXPUP),XERRUP(MAXPUP),XMAXUP(MAXPUP),LPRUP(MAXPUP)
```

IDBMUP: incoming beam particles (PDG codes, $\mathrm{p} = 2212$, $\overline{\mathrm{p}} = -2212$)

EBMUP: incoming beam energies (GeV)

PDFGUP, PDFSUP: PDFLIB parton distributions (not used by PYTHIA)

IDWTUP: weighting strategy

- = 1: PYTHIA mixes and unweights events, according to known $\mathrm{d}\sigma_{\mathrm{max}}$
- = 2: PYTHIA mixes and unweights events, according to known σ_{tot}
- = **3**: unit-weight events, given by user, always to be kept
- = 4: weighted events, given by user, always to be kept
- = -1, -2, -3, -4: also allow negative $\mathrm{d}\sigma$

NPRUP: number of separate user processes

XSECUP(i): σ_{tot} for each user process

XERRUP(i): error on σ_{tot} for each user process

XMAXUP(i): $\mathrm{d}\sigma_{\mathrm{max}}$ for each user process

LPRUP(i): integer identifier for each user process

The event

```
 INTEGER MAXNUP
 PARAMETER (MAXNUP=500)
 INTEGER NUP,IDPRUP,IDUP,ISTUP,MOTHUP,ICOLUP
 DOUBLE PRECISION XWGTUP,SCALUP,AQEDUP,AQCDUP,PUP,VTIMUP,SPINUP
 COMMON/HEPEUP/NUP,IDPRUP,XWGTUP,SCALUP,AQEDUP,AQCDUP,
&IDUP(MAXNUP),ISTUP(MAXNUP),MOTHUP(2,MAXNUP),ICOLUP(2,MAXNUP),
&PUP(5,MAXNUP),VTIMUP(MAXNUP),SPINUP(MAXNUP)
```

`IDPRUP`: identity of current process

`XWGTUP`: event weight (meaning depends on `IDWTUP` weighting strategy)

`SCALUP`: scale Q of parton distributions etc.

`AQEDUP`: α_{em} used in event

`AQCDUP`: α_{s} used in event

`NUP`: number of particles in event

`IDUP(i)`: PDG identity code for particle `i`

`ISTUP(i)`: status code ($-1 =$ incoming parton, $1 =$ final-state parton, $2 =$ intermediate resonance with preserved m)

`MOTHUP(j,i)`: position of one or two mothers

`PUP(j,i)`: (p_x, p_y, p_z, E, m)

`VTIMUP(i)`: invariant lifetime $c\tau$

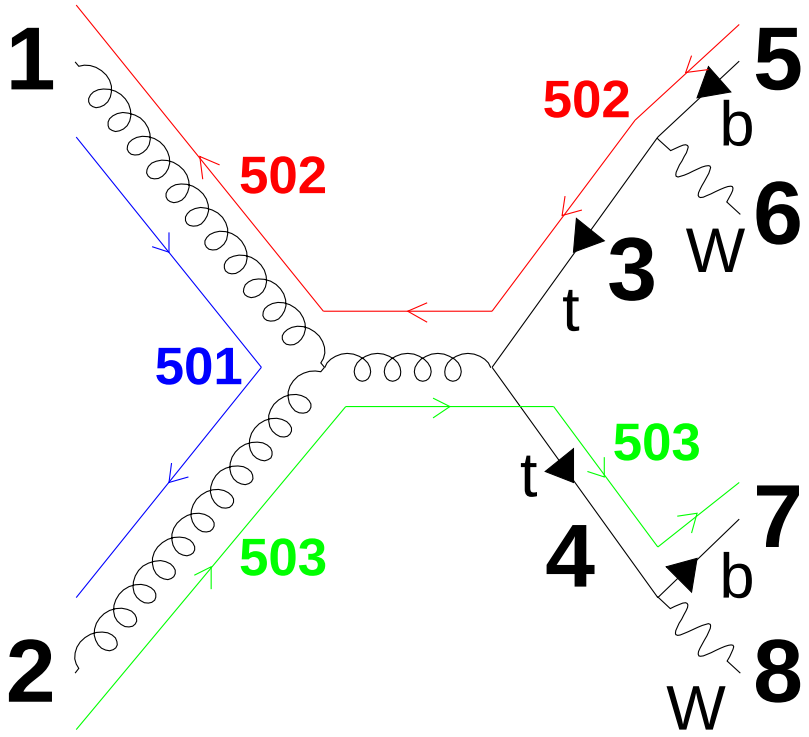
`SPINUP(i)`: spin (helicity) information

Examples of colour flows and indices

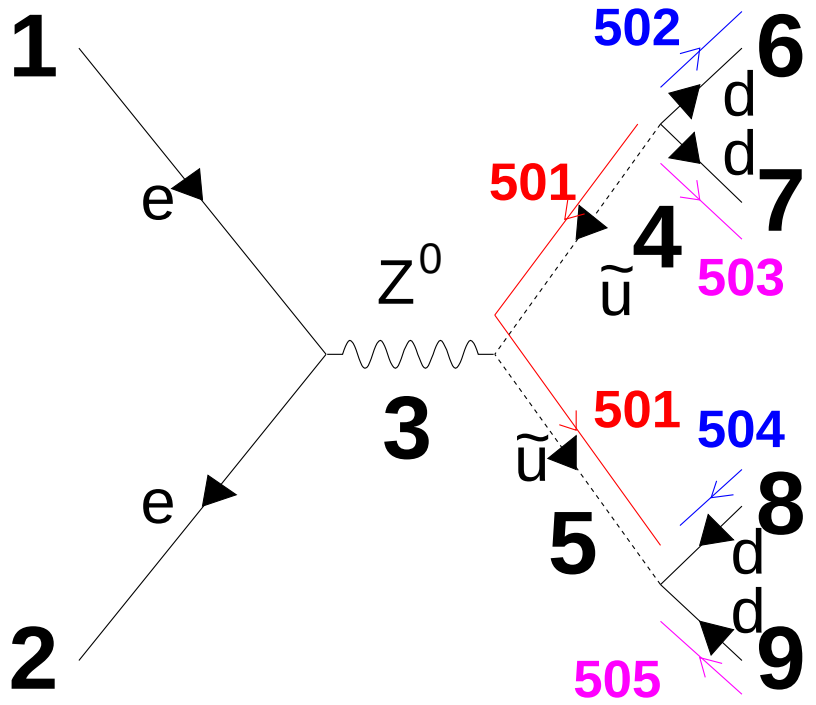
`ICOLUP(j,i)`: colour and anticolour indices

= colour line tags, in the $N_C \to \infty$ limit, starting e.g. with number 501.

Example 1: hadronic $t\bar{t}$ production



Example 2: baryon number violation



A C++ Implementation — Proposal For Discussion

Introduce two base classes:

- `LHAinit` : initialization info, $\sim$ `COMMON/HEPRUP/`
  pure virtual method `set`, $\sim$ `UPINIT`
- `LHAevnt` : event info, $\sim$ `COMMON/HEPEUP/`
  pure virtual method `set`, $\sim$ `UPEVNT`

The base classes provide

- methods for *extracting* all the Les Houches information,
- overloaded $<<$ for printing information, and
- the *tools* for storing information

Derived classes do the actual storing, with `set`, separately for

- external C++ process libraries
- reading from event file (MadGraph, AlpGen, . . .)
- interface to Fortran 77 commonblocks

LHAinit

Public methods:

`idBeamA(), idBeamB()`: incoming beam particles

`eBeamA(), eBeamB()`: incoming beam energies (GeV)

`pdfGroupBeamA(), pdfGroupBeamB(),`
`pdfSetBeamA(), pdfSetBeamB()`: PDF's

`strategy()`: weighting strategy

`size()`: number of processes, index `i` in range $0 \leq$ `i` $<$ `size`

`idProcess(i)`: integer identifier for each process

`xSec(i)`: σ_{tot} for each process

`xErr(i)`: error on σ_{tot} for each process

`xMax(i)`: $\mathrm{d}\sigma_{\mathrm{max}}$ for each process

Protected methods, to be used by `set`:

`LHAinit`, `~LHAinit`: constructor, destructor

`beamA(id, e, pdfGroup, pdfSet)`, same for `beamB`: set beams

`strategy(choice)`: set weighting strategy

`process(id, xSec, xErr, xMax)`: append process to list

LHAevnt

Public methods:

`idProc()`: identity of current process

`weight()`: event weight

`scale()`: scale Q of parton distributions etc.

`alphaQED()`, `alphaQCD()`: α_{em}, α_{s} used in event

`size()`: number of particles $+1$, index `i` in range $1 \leq$ `i` $<$ `size`
(keep slot 0 empty, for consistency with Fortran, mothers/daughters)

`id(i)`: PDG identity code for particle `i`

`status(i)`: status code

`mother1()`, `mother2()`: position of one or two mothers

`col1()`, `col2()`: colour and anticolour indices

`px(i)`, `py(i)`, `pz(i)`, `e(i)`, `m(i)`: (p_x, p_y, p_z, E, m)

`tau(i)`: invariant lifetime $c\tau$

`spin(i)`: spin (helicity) information

Protected methods, to be used by `set`:

`LHAevnt`, `~LHAevnt`: constructor, destructor

`process(id, weight, scale, alphaQED, alphaQCD)`: info on process

`particle(id, status, mother1, mother2, col1, col2,`
`px, py, pz, e, m, tau, spin)`: info on particle

Status

Up and running, used in PYTHIA 8

Roughly 500 lines, whereof $\sim$ half blank lines and comments.

Available on request (part of upcoming first PYTHIA 8 draft release).

Contains the two base classes, plus two derived class sets:

`LHAinitFortran, LHAevntFortran`: reads from Fortran commonblocks
used for runtime link to PYTHIA 6.3

`LHAinitPythia6, LHAevntPythia6`: reads from files
used for generation from stored PYTHIA 6.3 processes

```
LHAinitPythia6 lhaInit("sample.init");
LHAevntPythia6 lhaEvnt("sample.evnt");
pythia.init(&lhaInit, &lhaEvnt);
```

Still missing: derived classes for MadGraph, . . .

Outlook

The Les Houches Accord has been a big success,
influencing the way theorists structure event generators,
and the way experimentalists use them.

It could be *even more* useful if

- Further information were provided, e.g.
  - ⋆ Phase space cutoffs in ME generation
    e.g. for CKKW–L–MLM matching of ME and PS
  - ⋆ production scale of individual partons
    e.g. BFKL/CCFM gives ME+ISR, wants to add FSR & the rest
    (H. Jung, CASCADE)
- Initialization/event files had a standard format
  (S. Mrenna: MadGraph provides good example)

Should we start discussing an LHA++ ?