POWHEG with PYTHIA 8 in GAUSS

Philip Ilten

University College Dublin

Bucharest MC Workshop

November 22, 2012

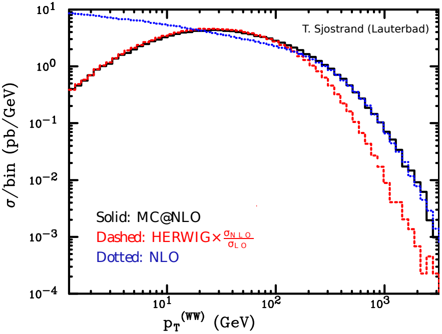
Overview

LO n-jet $\mathcal{ME} - \mathcal{PS}$ Merging

CKKW(L)	MLM
Catani, Krauss, Kuhn, Webber	Mangano, Moretti, Piccinini, Treccani
hep-ph/0109231	hep-ph/0611129
1. Calculate Sudakov factor an all lines. 2. Shower, reject emission using factor.	1. Perform shower and cluster jets. 2. Match jets to partons, reject if $N_p \neq N_{\text{jets}}$.
Sherpa	Alpgen/Herwig++
MadEvent/Pythia	MadEvent/Pythia

NLO $\mathcal{ME} - \mathcal{PS}$ Merging

Mc@Nlo	Powheg
Frixione, Webber	Frixione, Nason, Oleari
hep-ph/0402116	0709.2092



Mc@Nlo

1. Calculate NLO corrections to n-body process.
2. Calculate first shower emission.
3. Break event into NLO $\mathcal{ME}-$ first emission, remainder.
4. Apply showers to both parts of event.

Advantages:

- NLO variables
- smooth matching with $\mathcal{PS}$
- large number of processes

Disadvantages:

- negative weighting
- showering with Herwig(++) only

$pp \to (Z/\gamma^* \to) l_{\mathrm{IL}} \bar{l}_{\mathrm{IL}} + X$
$pp \to (W^+ \to) l^+_{\mathrm{IL}} \nu_{\mathrm{IL}} + X$
$pp \to Z^0 + X$
$pp \to H^0 + X$
$pp \to t/\bar{t} + X$
$pp \to tW^-/\bar{t}W^+ + X$
$pp \to tH^-/\bar{t}H^+ + X$
$pp \to H^0 W^+ + X$
$pp \to H^0 (W^- \to) l_i^- \bar{\nu}_i + X$
$pp \to W^+ W^- + X$
$pp \to W^- Z^0 + X$

$pp \to (Z \to) l_{\mathrm{IL}} \bar{l}_{\mathrm{IL}} + X$
$pp \to (W^- \to) l^-_{\mathrm{IL}} \bar{\nu}_{\mathrm{IL}} + X$
$pp \to W^+ + X$
$pp \to b\bar{b} + X$
$pp \to \bar{t} + X$
$pp \to \bar{t}W^+ + X$
$pp \to \bar{t}H^+ + X$
$pp \to H^0 (W^+ \to) l_i^+ \nu_i + X$
$pp \to H^0 Z + X$
$pp \to Z^0 Z^0 + X$

$pp \to (\gamma^* \to) l_{\mathrm{IL}} \bar{l}_{\mathrm{IL}} + X$
$pp \to \gamma^* (\to \sum_i f_i \bar{f}_i) + X$
$pp \to W^- + X$
$pp \to t\bar{t} + X$
$pp \to t + X$
$pp \to tW^- + X$
$pp \to tH^- + X$
$pp \to H^0 W^- + X$
$pp \to H^0 (Z \to) l_i \bar{l}_i + X$
$pp \to W^+ Z^0 + X$

Powheg

1. Pick largest p_T emission from NLO normalized $\mathcal{ME}$.
2. Evolve shower downwards to p_T scale.

Advantages:

- positive weights
- separation of shower

Disadvantages:

- designed for p_T ordered showers

Herwig++	PowhegBox	
$pp \to H$	$pp \to W$	$pp \to Z$
$pp \to HW$	$pp \to W + \text{jet}$	$pp \to Z + \text{jet}$
$pp \to ZH$	$pp \to t$	$pp \to tW$
$pp \to W$	$gg \to H$	$pp \to VV \to H$
$pp \to Z$	$pp \to \text{jet} + \text{jet}$	$pp \to t\bar{t}$
	$pp \to WW + \text{dijet}$	$pp \to WW$
	$pp \to WZ$	$pp \to ZZ$
	$pp \to b\bar{b}WW$	

PowhegBox with Pythia 8

Shower Interface

- Already done in `main31.cc` (partially) by Richard Corke.
- Needs adjustment per matrix-element.

```
pythia.readString("SpaceShower:→
    pTmaxMatch = 2");
pythia.readString("TimeShower:→
    pTmaxMatch = 2");
pythia.readString("→
    MultipartonInteractions:→
    pTmaxMatch = 2");

powhegHooks = new PowhegHooks(→
    nFinal, vetoMode, vetoCount,
                pThardMode, pTemtMode, →
                    emittedMode,
                pTdefMode, MPIvetoMode)→
                    ;
pythia.setUserHooksPtr((UserHooks →
    *) powhegHooks);
```

Technical Interface

- Experimental interface in `Gauss/Gen/LbPowheg`.
- Common structure between libraries.
    - Degenerate names.
- Input settings from file.
    - Stores initialization in files.
    - PDF from file.
- Event passed to shower through common blocks.
- Internal random number generation.

Algorithm

1. Find p_T scale during multiple interaction phase.
   - If explicit radiation in record, set as p_T scale.
   - Otherwise set as event momentum fraction as p_T scale.
2. Veto if first ISR emission above p_T scale.
3. Veto if first FSR emission above p_T scale.

Shower Hooks

- Interface to shower through `UserHooks` with 6 access points to generation process.
  - Use `doVetoMIStep` to find p_T scale.
  - Use `doVetoISREmission` to veto ISR.
  - Use `doVetoFSREmission` to veto FSR.

```
class PowhegHooks : public UserHooks {
  // Determine the p_T scale.
  bool canVetoMIStep()    { return true; }
  int  numberVetoMIStep() { return 1; }
  bool doVetoMIStep(int, const Event &e) {
      pTveto   = infoPtr->QFac();
      pTveto = pTpowheg = e[6].pT();
      return false;
  }

  // Veto both ISR and FSR emissions above the p_T scale.
  bool canVetoISREmission() { return true; }
  bool doVetoISREmission(int, const Event &e) {
    // Return accordingly.
  }
};
```

Example

- Veto dependent on hard process structure.
- Need to supply correct veto per PowhegBox matrix element.

```
 --------  PYTHIA Event Listing  (complete event)  -----------------

    no    id    name      status  mothers   daughters     e        m
     0    90   (system)     -11   0     0    0     0 7000.000 7000.000
     1  2212   (p+)         -12   0     0    3     0 3500.000    0.938
     2  2212   (p+)         -12   0     0    4     0 3500.000    0.938
     3    -1   (dbar)       -21   1     0    5     6    7.369    0.000
     4     2   (u)          -21   2     0    5     6  339.234    0.000
     5    24   W+            22   3     4    0     0  313.409   92.718
     6    21   g             23   3     4    0     0   33.194    0.000

 --------  End PYTHIA Event Listing  ------------------------------
```

Gauss Package

- `Gauss/Gen/LbPowheg`
  - Based on `LbPythia8`.

```
#==========================================
use Generators v* Gen
use pythia8    v* →
     LCG_GeneratorsInterfaces

#==========================================
library LbPowheg_w Lib/dijet/*.[fF]→
      Lib/dijet/*.cxx
...
library LbPowhegLib Lib/*.cpp Lib→
     /*.cxx
library LbPowheg component/*.cpp

#==========================================
 macro_append fflags "-fno-→
      automatic -fPIC -ffixed-line-→
      length-none -fno-second-→
      underscore -O2"
```

- `cmt`
  - `package.sh` - parses PowhegBox source
- `src/component`
  - `PowhegProduction` - actual production source
- `src/Lib`
  - `GaudiRandomForPowheg` - Pythia random numbers
  - `PowhegHooks` - shower hooks for elements
  - `powheg` - wrapper to PowhegBox
  - `<process>` - Powheg $\mathcal{ME}$ source

Common Structure

- Ideally, different POWHEGBOX structure.
- Instead, name mangling with SED (poor-man's lexical analyzer).
- Implemented in `cmt/package.sh`.

```
# Convert library name to all lower case.
# This needs to be done because GCC assigns the external symbol names in
# all lower case, so all lower case is used for consistancy.
LIB_LC=`echo "${LIB}" | tr "[:upper:]" "[:lower:]"`

# Fix the include file paths.
sed -i "s/\([[:space:]]\+include[[:space:]]\+['\"]\)[^'\"]*\//\1/gi" $TRGDIR→
    /$LIB_LC/*

# Find all subroutines.
SUBROUTINES=`grep --no-filename --ignore-case --only-matching "^[[:space:]]*→
    subroutine[[:space:]]*[[:alnum:]_]*" $TRGDIR/$LIB_LC/* | sed 's/→
    subroutine//gi' `

# Find all entries.
# Find all functions.
# Find all common blocks.
# Find all data blocks.

# Mangle the names for all subroutines, functions, and common blocks.
mangle_names $ENTRIES $SUBROUTINES $FUNCTIONS $COMMONS $DATAS
```

Input

- Reads input settings, PDF's from files in current directory.
- Ideally, pass as strings, but requires POWHEGBOX changes.
- Read in through C++, parse, write to temporary input and PDF files.
- Implemented in `src/Lib/powheg.cxx`.

```
// Open the output configuration file.
fstream config("powheg.input", ios::out);

// Copy the settings to the configuration file.
for (unsigned int i = 0; i < settings.size(); i++) {
  config << settings[i] << "\n";
}
config.close();

// Open the input and output PDF files.
fstream pdfin(pdf.c_str(), ios::in | ios::binary);
fstream pdfout("cteq6m", ios::out | ios::binary);

// Copy the PDF input to the PDF output.
pdfout << pdfin.rdbuf(); pdfin.close(); pdfout.close();
```

Event Interface

- Variables stored in Les Houches common blocks.
- Access through external hook and read into Pythia 8.
- Implemented in `src/Lib/<process>/<process>.cxx` for parameters, event, and random numbers.

```
// External FORTRAN hooks to POWHEG.
extern "C" {

  // The event Les Houches common block.
  extern struct {
    int nup, idprup;
    double xwgtup, scalup, aqedup, aqcdup;
    int idup[500], istup[500], mothup[500][2], icolup[500][2];
    double pup[500][5], vtimup[500],spinup[500];
  } LIB_hepeup_;
}

// Store particle info.
for (int ip = 0; ip < LIB_hepeup_.nup; ++ip)
  addParticle(LIB_hepeup_.idup[ip], LIB_hepeup_.istup[ip],
        LIB_hepeup_.mothup[ip][0], LIB_hepeup_.mothup[ip][1],
        LIB_hepeup_.icolup[ip][0], LIB_hepeup_.icolup[ip][1],
        LIB_hepeup_.pup[ip][0], LIB_hepeup_.pup[ip][1],
        LIB_hepeup_.pup[ip][2], LIB_hepeup_.pup[ip][3],
        LIB_hepeup_.pup[ip][4], LIB_hepeup_.vtimup[ip],
        LIB_hepeup_.spinup[ip]);
```

Conclusion

- Experimental PowhegBox with Pythia 8 package now in Gauss.
- Shower interface needs further work and validation.
- Random number interface needed for PowhegBox.
- What are the requested processes?
- What timeline is necessary?
- Is dedicated interface necessary? Just pass HepMC?
- Is Mc@Nlo with Herwig++ more desirable?