status of APPLfast interpolation grids for NNLO

DIS18, Kobe, Japan, April 2018

D. Britzger (Heidelberg)
C. Gwenlan (Oxford)
A. Huss (CERN)
K. Rabbertz (KIT)
M. Sutton (Sussex)

with support from
motivation, and APPLfast

- interpretation of exp. data requires fast theory predictions
  often need repeated computation of same cross section, for EG: pdf uncertainties and/or alternative sets; scale variations, $\mu_R, \mu_F$; variation of $\alpha_s(M_Z)$; SM parameter fits

- jet cross section calcs. at NLO were slow – historical reason for development of interpolation grids
  - nowadays NNLO in general very demanding!
  - need procedure for fast repeated computations of higher order cross sections
    $\rightarrow$ interpolation grids using APPLgrid or fastNLO

- APPLfast: common project of APPLgrid, fastNLO and NNLOJET authors:
  - NNLOJET: semi-automated calculation of NNLO QCD cross sections
    (authors from CERN, ETH, Zurich, IPPP, Lisbon)
  - APPLfast: interface between NNLOJET and fast grid technology:
    - implementation for both APPLgrid and fastNLO;
    - aims to be as unobtrusive as possible, for both ends of interface;
    - flexible; intended to be reusable by other theory codes

(see also write up in Les Houches proceedings, arXiv:1803.07977)
APPLfast grid generation (workflow)

1. **pre-processing**: check of interpolation quality, and optimise if necessary (e.g. number of grid nodes, interpolation order etc.) \(O(10 \text{ h})\)

2. **NNLOJET warm-up**: optimise the NNLOJET VEGAS phase space in dedicated NNLOJET job [1 long (multicore) job per process] \(O(100 \text{ h})\)

3. **NNLOJET – APPLfast warmup**: run with grid filling enabled to establish optimised phase space; exact strategy differs between APPLgrid and fastNLO, but this is hidden in interface; only phase space provided by NNLOJET, providing significant speed up \(O(100 \text{ h})\)

4. **grid production run**: thousands of jobs in parallel \(O(250 \text{ kh})\)

5. **post-processing**: statistical evaluation and combination of all produced output sub-grids from production run \(O(100 \text{ h})\)

6. **validation, validation, validation** \(O(?? \text{ h})\)

7. **present final results** 30 mins!
step 1: pre-processing

Z+Jet approximation test jobs

note y-axis range; sub-permille agreement reached in LO, NLO and NNLO in validation jobs
step 2: initial VEGAS warm up

NNLOJET warmup – no grid generation

**one job** per cross section contribution type EG. LO, R, V, RR(a,b), RV, VV;

internal NNLOJET multi-threading possible

<table>
<thead>
<tr>
<th>Job Type</th>
<th># Jobs</th>
<th>Threads / Job</th>
<th>Events / Job</th>
<th>Runtime / Job</th>
<th>Total Runtime</th>
</tr>
</thead>
<tbody>
<tr>
<td>LO</td>
<td>1</td>
<td>16</td>
<td>32 M</td>
<td>0.35 h</td>
<td>0.35 h</td>
</tr>
<tr>
<td>NLO-R</td>
<td>1</td>
<td>16</td>
<td>16 M</td>
<td>1.0 h</td>
<td>1.0 h</td>
</tr>
<tr>
<td>NLO-V</td>
<td>1</td>
<td>16</td>
<td>16 M</td>
<td>1.0 h</td>
<td>1.0 h</td>
</tr>
<tr>
<td>NNLO-RRa</td>
<td>1</td>
<td>32</td>
<td>5 M</td>
<td>17.5 h</td>
<td>17.5 h</td>
</tr>
<tr>
<td>NNLO-RRb</td>
<td>1</td>
<td>32</td>
<td>5 M</td>
<td>20.7 h</td>
<td>20.7 h</td>
</tr>
<tr>
<td>NNLO-RV</td>
<td>1</td>
<td>16</td>
<td>8 M</td>
<td>22.4 h</td>
<td>22.4 h</td>
</tr>
<tr>
<td>NNLO-VV</td>
<td>1</td>
<td>16</td>
<td>8 M</td>
<td>24.6 h</td>
<td>24.6 h</td>
</tr>
<tr>
<td><strong>Total</strong></td>
<td>7</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>87.6 h</td>
</tr>
</tbody>
</table>

(NB, a/b indicates a technical phase space separation for RR)
**step 3: phase space exploration**

**APPLfast warmup:**

NNLOJET run without CPU-time expensive weight calculation;

at least one job per process needed to determine phase space limits individually;

jobs can be parallelised if necessary

<table>
<thead>
<tr>
<th>Job Type</th>
<th># Jobs</th>
<th>Events / Job</th>
<th>Runtime / Job</th>
<th># Events</th>
<th>Total Runtime</th>
</tr>
</thead>
<tbody>
<tr>
<td>LO</td>
<td>5</td>
<td>500 M</td>
<td>12 h</td>
<td>2.5 G</td>
<td>60 h</td>
</tr>
<tr>
<td>NLO-R</td>
<td>5</td>
<td>300 M</td>
<td>18 h</td>
<td>1.5 G</td>
<td>90 h</td>
</tr>
<tr>
<td>NLO-V</td>
<td>5</td>
<td>500 M</td>
<td>13 h</td>
<td>2.5 G</td>
<td>65 h</td>
</tr>
<tr>
<td>NNLO-RRa</td>
<td>10</td>
<td>50 M</td>
<td>13 h</td>
<td>0.5 G</td>
<td>130 h</td>
</tr>
<tr>
<td>NNLO-RRb</td>
<td>10</td>
<td>50 M</td>
<td>15 h</td>
<td>0.5 G</td>
<td>150 h</td>
</tr>
<tr>
<td>NNLO-RV</td>
<td>5</td>
<td>300 M</td>
<td>19 h</td>
<td>1.5 G</td>
<td>90 h</td>
</tr>
<tr>
<td>NNLO-VV</td>
<td>5</td>
<td>500 M</td>
<td>12 h</td>
<td>2.5 G</td>
<td>60 h</td>
</tr>
<tr>
<td><strong>Total</strong></td>
<td><strong>45</strong></td>
<td><strong>---</strong></td>
<td><strong>---</strong></td>
<td><strong>11.5 G</strong></td>
<td><strong>645 h</strong></td>
</tr>
</tbody>
</table>

(presented tables used for extensive testing; overkill for normal use)
step 4: mass production

**NNLOJET+APPLfast:**
massive parallelised computing on virtual machines with 24h lifetime;

**example with fastNLO:**

<table>
<thead>
<tr>
<th>Job Type</th>
<th># Jobs</th>
<th>Events / Job</th>
<th>Runtime / Job</th>
<th># Events</th>
<th>Total Output</th>
<th>Total Runtime</th>
</tr>
</thead>
<tbody>
<tr>
<td>LO</td>
<td>10</td>
<td>140 M</td>
<td>20.6 h</td>
<td>1.4 G</td>
<td>24 MB</td>
<td>206 h</td>
</tr>
<tr>
<td>NLO-R</td>
<td>200</td>
<td>6 M</td>
<td>19.0 h</td>
<td>1.2 G</td>
<td>1.3 GB</td>
<td>3800 h</td>
</tr>
<tr>
<td>NLO-V</td>
<td>200</td>
<td>5 M</td>
<td>21.2 h</td>
<td>1.0 G</td>
<td>1.2 GB</td>
<td>4240 h</td>
</tr>
<tr>
<td>NNLO-RRa</td>
<td>5000</td>
<td>60 k</td>
<td>22.5 h</td>
<td>0.3 G</td>
<td>26 GB</td>
<td>112500 h</td>
</tr>
<tr>
<td>NNLO-RRb</td>
<td>5000</td>
<td>40 k</td>
<td>20.3 h</td>
<td>0.2 G</td>
<td>27 GB</td>
<td>101500 h</td>
</tr>
<tr>
<td>NNLO-RV</td>
<td>1000</td>
<td>200 k</td>
<td>19.8 h</td>
<td>0.2 G</td>
<td>6.4 GB</td>
<td>19800 h</td>
</tr>
<tr>
<td>NNLO-VV</td>
<td>300</td>
<td>4 M</td>
<td>20.5 h</td>
<td>1.2 G</td>
<td>2.0 GB</td>
<td>6150 h</td>
</tr>
<tr>
<td>Total</td>
<td>11710</td>
<td>---</td>
<td>---</td>
<td>5.5 G</td>
<td>64 GB</td>
<td>248196 h</td>
</tr>
</tbody>
</table>

3 × 11710 grids/tables + all NNLOJET output!
Final 3 files for analysis are O(10MB) each

NB, roughly factor of 2 (only!) penalty for running with grid production vs NNLOJET alone
(depends in detail on physics process and grid set up)
step 5: post-processing

- checking, purging, combining
- check again interpolation quality for individual grids
- run NNLOJET combination script $\rightarrow$ weight tables
- weighted merging of grids
- check and treat any potential remaining outliers
- … do some interesting physics
step 6: validation

- check every aspect we can think of… else, Murphy’s Law!
- check each contribution (LO, R, V, RR(a,b), RV, VV) separately
- check interpolation in x-space for single grids
- check interpolation in scales for single grids
- compare merged grids to NNLOJET for each contribution
- compare final merged grids for each order to NNLOJET
- more checks and comparisons EG. to other programs
- ETC ETC.
inclusive jet pt – single grid

+10%  
\text{ratio} = \text{APPLfast}/\text{NNLOJET}

\textbf{error bars:}
stat. uncertainty estimate from NNLOJET

±1‰

-10%
inclusive jet pt – single grid
inclusive jet pt – combined grid

closure quality of APPLfast

error bars: statistical uncertainty estimate from NNLOJET
inclusive jet pt – combined grid

Merged grid: 1jet RRa–CMS7 fnl2332d_xptji_y1 for scale choice \( p_{T,\text{jet}} \)

-50%

-50%

+5%

+5%

Merged grid: 1jet RRb–CMS7 fnl2332d_xptji_y1 for scale choice \( p_{T,\text{jet}} \)

+50%

-50%

+5%

+5%

status as of Monday PM

Merged grid: 1jet RV–CMS7 fnl2332d_xptji_y1 for scale choice \( p_{T,\text{jet}} \)

+50%

-50%

-5%

-5%

Merged grid: 1jet VV–CMS7 fnl2332d_xptji_y1 for scale choice \( p_{T,\text{jet}} \)

+50%

-50%

-5%

-5%
inclusive jet pt – combined grid

closure quality of \texttt{APPLfast}

\textbf{error bars:} statistical uncertainty estimate from NNLOJET

status as of Monday PM
cross check with NLOJet++

ratio always to NNLOJET with scale ptmax

error bars: stat. uncertainty estimate from NNLOJET and NLOJet++
cross check with NLOJet++

ratio always to NNLOJET with scale ptmax

error bars: stat. uncertainty estimate from NNLOJET and NLOJet++
**pre-step 7: results**

CMS inclusive jet cross section, APPLfast vs NNLOJET

**DISCLAIMER:** currently relatively poor stats for some NNLO contributions; required precision only at LO and NLO – only representative cross section, not yet for physics use.
LO, NLO and NNLO cross sections

**DISCLAIMER:** currently relatively poor stats for some NNLO contributions; required precision only at LO and NLO – only representative cross section, not yet for physics use
outlook

NNLOJET provides NNLO QCD calculations in common interface
- **pp:** Z & W incl., Z+jet, W+jet, jet+dijet, H incl., H+jet; **ep:** jet+dijet; **ee:** 3jets

**APPLfast** interface (NNLO-bridge) is working
numerous adaptions implemented by all sides for optimal performance

large scale productions tested for **pp** Z+jet, **pp** inclusive jet and **DIS** jet

final combination prescription for NNLOJET results received last Nov.
removes fluctuations from incomplete cancellations, weighted interpolation table merging implemented

production that was launched for CMS **pp** incl. jets at 7 TeV recently finished;
result shown here for first time

many new NNLO interpolation grids planned
grid distribution – Ploughshare

Ploughshare
for all your interpolation grid needs

Ploughshare allows users from the HEP community to share fast interpolation grids in a standardised way. PDF fitters and those from the experimental collaborations are be able to upload their validated grids and access the grids of others quickly and with the minimum of fuss.
grid distribution – Ploughshare

What is Ploughshare?

1. **Quick to use** - a web-based utility for the automated distribution of fast interpolation grids for the high energy physics community.

2. **Secure storage** - registered users can upload grid files and corresponding standard format configuration files to describe the grids and physics processes and these are added to a central repository.

3. **Automatic distribution** - a standard utility library will be provided to download any required grids automatically in user code.

A utility for the community: Ploughshare allows users to share their grids, so it is important that the provenance of the grids is guaranteed. This is achieved by allowing only registered users to upload their validated grids. Subsequently, however, anyone is free to download and use the grids as they wish.

Fast operations summary

Navigate quickly to some of the primary operations you might be interested in:

- **Download grids**: View all the lovely grids which are available for download.
- **Upload grids**: Upload grids using the standard web interface.
- **Download grid code**: Get the code for the automated download of multiple grids.
- **Settings**: How to set up the automated code for the grid downloads.

- **New project**: registered users can upload grids with documentation.

- **Automated job treats upload**
  - adds to appropriate location in file system
  - generates relevant lists and display web pages

- Provides user interface for **automated download** with a simple line of code

- Expression of interest from many stakeholders…

- Proof of concept ready…

**CONTRIBUTIONS WELCOME!**
summary

APPLfast interface (NNLO–Bridge) and interpolation is working large scale productions tested for pp Z+jet, pp inclusive jet and DIS jet combination of grids with weights à la NNLOJET implemented addressing last issues, checking on possible remaining outliers in grids; finalising validation starting to produce a series of APPLgrid and/or fastNLO tables for various processes with publically available data final grids will be made available via a common repository; open for contributions from the community
extras
production campaign

optimised scenario: finished in two days with 7800 parallel jobs at maximum

(thanks to bwHPC and the NEMO HPC cluster team in Freiburg)
inclusive jet pt – combined grid

Merged grid: 1jet RRa–CMS7 fnl2332d_xptji_y1 for scale choice $p_T^{\text{max}}$

Merged grid: 1jet RRb–CMS7 fnl2332d_xptji_y1 for scale choice $p_T^{\text{max}}$

status as of Monday PM
grid combination Z+Jets

Grid combination

- Following the production, the grids need to be combined
  - Alex and Tom’s cross section combination code weights individual bins for the cross sections
  - Can combine the grids weighting the different bins in a consistent manner …
- Procedure works

NNLOJET combination
Simply added grids

Z+Jet VV contribution
ratio APPLfast / NNLOJET

NNLOJET combination
Simply added grids

Z+Jet VV contribution
ratio of APPLfast convolution to NNLOJET combination reference

M Sutton - APPLfast-NNLO
cross section comparison (LO)

LO CMS inclusive jet cross section, NNLOJET vs APPLfast

Stat. uncertainty & grid closure
- NNLOJET ± Δ_{stat}
- APPLfast grid (σ > 0)
cross section comparison (NLO)

NLO CMS inclusive jet cross section, NNLOJET vs APPLfast
Recap of the Numerical Technique

- For a calculation of a cross section from $m = 1 \ldots N$ weights, $w_m$, from a Monte Carlo integration with momentum fraction $x_m$, form the product

$$\sum_m w(x_m)q(x_m),$$

- Can interpolate the function $q(x_m)$

$$q(x_m) \approx \sum_i q^{(i)}I^{(i)}(x_m - x^{(i)}),$$

- such that

$$\sum_m w(x_m)q(x_m) \approx \sum_i q^{(i)} \sum_m w(x_m)I^{(i)}(x_m - x^{(i)})$$

$$\approx \sum_i q^{(i)}W^{(i)}$$

- For a calculation of a cross section with $m = 1 \ldots N$ weights, from a Monte Carlo integration with momentum transfer $Q^2$

$$d\sigma = \sum_p \sum_{m=1}^N w_m^{(p)} \left( \frac{\alpha_s(Q_m^2)}{2\pi} \right)^p q(x_m, Q_m^2)$$

$$= \sum_p \sum_{ij} q(x_{(i)}, Q_{(j)}^2) \left( \frac{\alpha_s(Q_{(j)}^2)}{2\pi} \right)^p \sum_m w_m^{(p)} I_i^x(x_m)I_j^2(Q_m^2)$$

$$= \sum_p \sum_{ij} q(x_{(i)}, Q_{(j)}^2) \left( \frac{\alpha_s(Q_{(j)}^2)}{2\pi} \right)^p W_{ij}^{(p)}$$
For \( pp \) collisions need an extra dimension for the PDF of the second colliding hadron

\[
d\sigma = \sum_p \sum_{m=1}^N w_m^{(p)} \left( \frac{\alpha_s(Q^2_m)}{2\pi} \right)^p q_1(x_{1m}, Q^2_m) q_2(x_{2m}, Q^2_m)
\]

But there is an implicit summation over parton flavours. Make use of symmetries in the matrix elements to use a vector of \( k = 1 \ldots M \) independent weights such that

\[
\sum_{ij=q,\bar{q},g} w_{ij} q_{1i}(x_1) q_{2j}(x_2) = \sum_{k=1}^M w^{(k)} F^{(k)}(x_1, x_2)
\]

so that

\[
d\sigma = \sum_p \sum_{k=1}^M \sum_{m=1}^N w_m^{(p)(k)} \left( \frac{\alpha_s(Q^2_m)}{2\pi} \right)^p F_m^{(k)}(x_{1m}, x_{2m}, Q^2_m)
\]

Which can be placed on a grid in the same way as for DIS

So from the summation, everything is down to the quality of the interpolation of the pdf at the grid nodes

- It is a pure quadrature technique and is not, in principle subject to statistical fluctuation, or put another way …
- Each individual weight gets added to the grid, and should be well approximated individually
Z+Jet NLO subprocesses

- **NNLOJET** calculates many (150) distinct internal processes, many with same input partons
- automatically reduce to 33 parton luminosities for NLO (keep internal mapping of internal process ID to parton luminosity)

\[
\begin{align*}
0 & : 13 27 41 42 65 73 103 133 & (d, d) + (s, s) + (b, b) \\
1 & : 14 28 43 44 66 74 104 134 & (u, u) + (c, c) \\
2 & : 15 29 45 46 67 75 105 135 & (d, d) + (s, s) + (b, b) \\
3 & : 16 30 47 48 68 76 106 136 & (u, u) + (c, c) \\
4 & : 17 31 77 107 137 & (d, g) + (s, g) + (b, g) \\
5 & : 18 32 78 108 138 & (u, g) + (c, g) \\
6 & : 19 33 79 109 139 & (g, d) + (g, s) + (g, b) \\
7 & : 20 34 80 110 140 & (g, u) + (g, c) \\
8 & : 21 22 35 36 85 86 115 116 145 146 & (g, g) \\
9 & : 23 37 81 111 141 & (g, d) + (g, s) + (g, b) \\
10 & : 24 38 82 112 142 & (g, u) + (g, c) \\
11 & : 25 39 83 113 143 & (d, g) + (s, g) + (b, g) \\
12 & : 26 40 84 114 144 & (u, g) + (c, g) \\
13 & : 49 91 121 151 & (d, d) + (d, s) + (d, b) + (s, d) + (s, s) + (b, s) + (b, b) \\
14 & : 50 92 122 152 & (d, u) + (d, c) + (s, u) + (s, c) + (b, u) + (b, c) \\
15 & : 51 93 123 153 & (u, d) + (u, s) + (u, b) + (c, d) + (c, s) + (c, b) \\
16 & : 52 94 124 154 & (u, u) + (u, c) + (c, u) + (c, c) \\
17 & : 53 87 117 147 & (d, d) + (d, s) + (d, b) + (s, d) + (s, s) + (s, b) + (b, d) + (b, s) + (b, b) \\
18 & : 54 88 118 148 & (d, u) + (d, c) + (s, u) + (s, c) + (b, u) + (b, c) \\
19 & : 55 89 119 149 & (u, d) + (u, s) + (u, b) + (c, d) + (c, s) + (c, b) \\
20 & : 56 90 120 150 & (u, u) + (u, c) + (c, u) + (c, c) \\
21 & : 57 99 129 159 & (d, d) + (d, s) + (d, b) + (s, d) + (s, s) + (s, b) + (b, d) + (b, s) + (b, b) \\
22 & : 58 100 130 160 & (d, u) + (d, c) + (s, u) + (s, c) + (b, u) + (b, c) \\
23 & : 59 101 131 161 & (u, d) + (u, s) + (u, b) + (c, d) + (c, s) + (c, b) \\
24 & : 60 102 132 162 & (u, u) + (u, c) + (c, u) + (c, c) \\
25 & : 61 95 125 155 & (d, d) + (d, s) + (d, b) + (s, d) + (s, s) + (s, b) + (b, d) + (b, s) + (b, b) \\
26 & : 62 96 126 156 & (d, u) + (d, c) + (s, u) + (s, c) + (b, u) + (b, c) \\
27 & : 63 97 127 157 & (u, d) + (u, s) + (u, b) + (c, d) + (c, s) + (c, b) \\
28 & : 64 98 128 158 & (u, u) + (u, c) + (c, u) + (c, c) \\
29 & : 69 & (d, d) + (s, s) + (b, b) \\
30 & : 70 & (u, u) + (c, c) \\
31 & : 71 & (d, d) + (s, s) + (b, b) \\
32 & : 72 & (u, u) + (c, c)
\end{align*}
\]
Z+Jet NNLO subprocesses

• many more individual internal processes (794); can be reduced down to same 33 parton luminosities
APPLfast grid filling developments

• for the grid filling, call the fill methods for each process generated
  – at NNLO nearly 800 separate processes, each called many times for each phase space point
  – calling fill methods for each weight is maximally time consuming

• thus, implemented a filling cache:
  – each weight added to a weight vector corresponding to a unique phase space point and observable value
  – when phase space point changes, flush cache to grid
  – significantly reduces number of calls to grid filling, by factors of 20–200, depending on process
### typical sample processing times

- **NNLOJET Z+Jet** without APPLfast grids
  - LO : 100 jobs × (5 mins)
  - NLO-V : 100 jobs × (10 mins)
  - NLO-R : 100 jobs × (15 mins)
  - NNLO-VV : 100 jobs × (1 h)
  - NNLO-RV : 1000 jobs × (10–20 h)
  - NNLO-RR : 5000 jobs × (20 h)

- **NNLOJET DIS jets** with grids
  - each job 8–16 h CPU time
  - LO : 50 jobs (5G events)
  - NLO-V : 40 jobs (2G events)
  - NLO-R : 80 jobs (2G events)
  - NNLO-VV : 100 jobs (1.5G events)
  - NNLO-RV : 5000 jobs (5G events)
  - NNLO-RRa : 10000 jobs (5G events)
  - NNLO-RRb : 2000 jobs (5G events)

  - initial performance penalty of grid production versus NNLOJET alone, reduced to roughly a factor of only about 2 (depends in detail on physics process and grid setup)

- still gain 100k’s of CPU hours for each avoided full run of calculation
Figure 1: Relative change of jet cross section as a function of a multiplicative factor applied to the renormalisation and factorisation scale for four exemplary data points of the HERA-II phase space. The bin definitions are displayed in the respective panels. The left panels show inclusive jet cross sections, and the right panels dijet cross sections. The full line shows the cross section dependence for the NNLO, the dashed line for NLO and the dotted line for LO calculations. For better comparison, all calculations are performed with the same PDF set (NNPDF3.1 NNLO).

For all panels, the cross sections are normalised to the respective NLO cross section with unity scale factor. The filled area around the NNLO calculation indicates variations of the factorisation scale by factors of 0.5 and 2 around the chosen value for $\mu_R$.

- DIS jets also available in NNLOJET (arXiv:1703.05977, 1606.03991)
- APPLfast interface functional and published for ep

used to extract value of $\alpha_s$ at NNLO using H1 inclusive and dijets in DIS (arXiv:1709.07251); grids also being produced and finalised for ZEUS