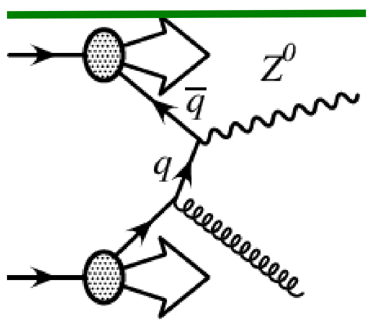
# Update on Interpolation Grids for NNLO

A. Huss (ETH Zürich), T. Morgan (IPPP Durham)
C. Gwenlan (Oxford), M. Sutton (Sussex)
D. Britzger (DESY), **K. Rabbertz (KIT)**

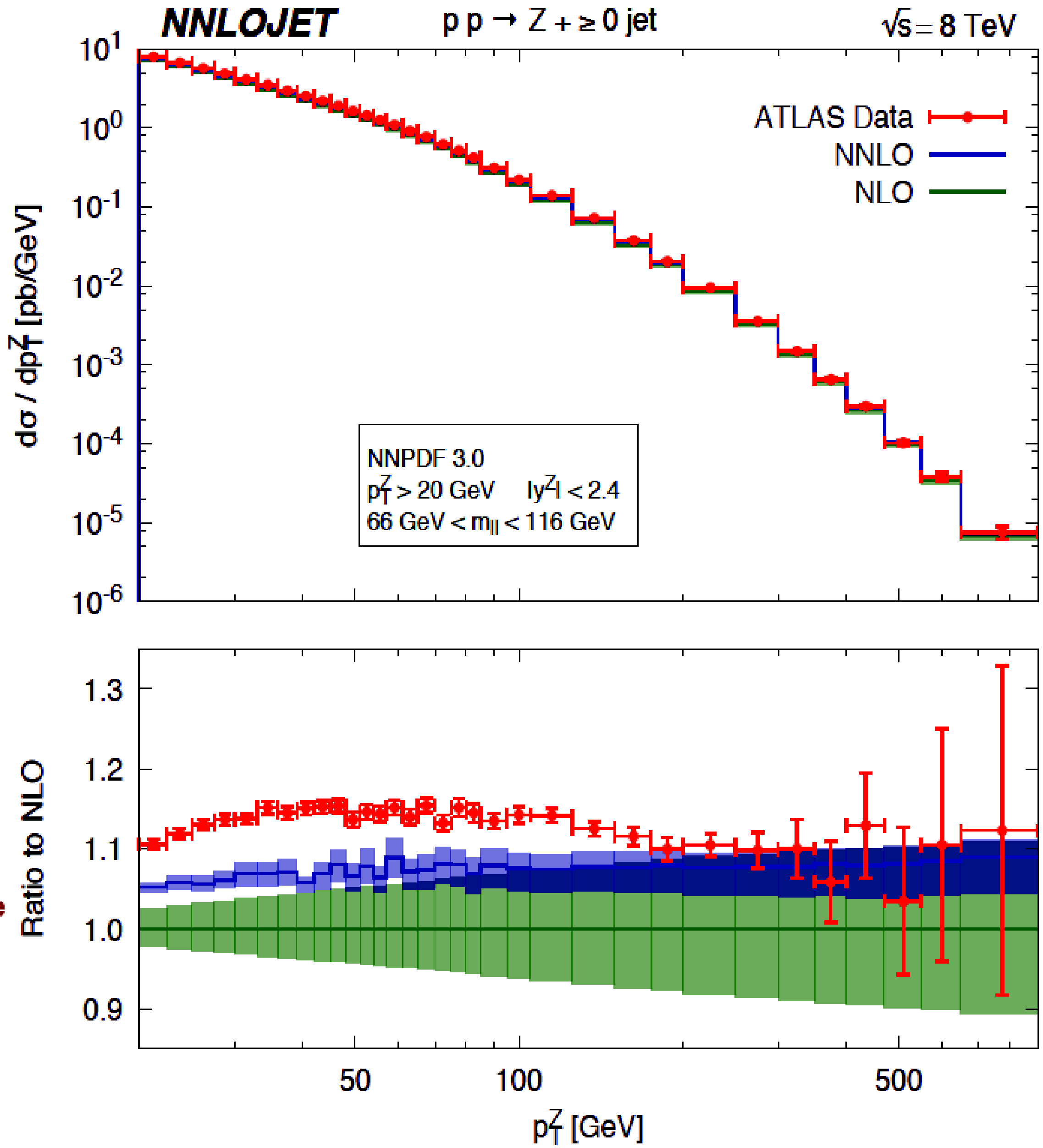
**With support from an IPPP Associateship**

# NNLOJET (and APPLfast-NNLO)

- Semi-automated calculation of cross sections at NNLO from the IPPP, Zurich, ETH and others
  - Gehrmann-De Ridder *et al* **arXiv: 1607.01749**
  - See talk from **Alex Huss** tomorrow
- APPLfast-NNLO
  - Developers from **NNLOJET**, **APPLgrid** and **fastNLO**
  - A single, combined interface for NNLOJET with both APPLgrid and fastNLO




- Many processes implemented in NNLOJET
  - Developing a generic interface for **all available** processes
  - Concentrating on Z + jets at NNLO for the initial development and proof-of-concept




Gehrmann et al., JHEP, 2016, 07, 133.

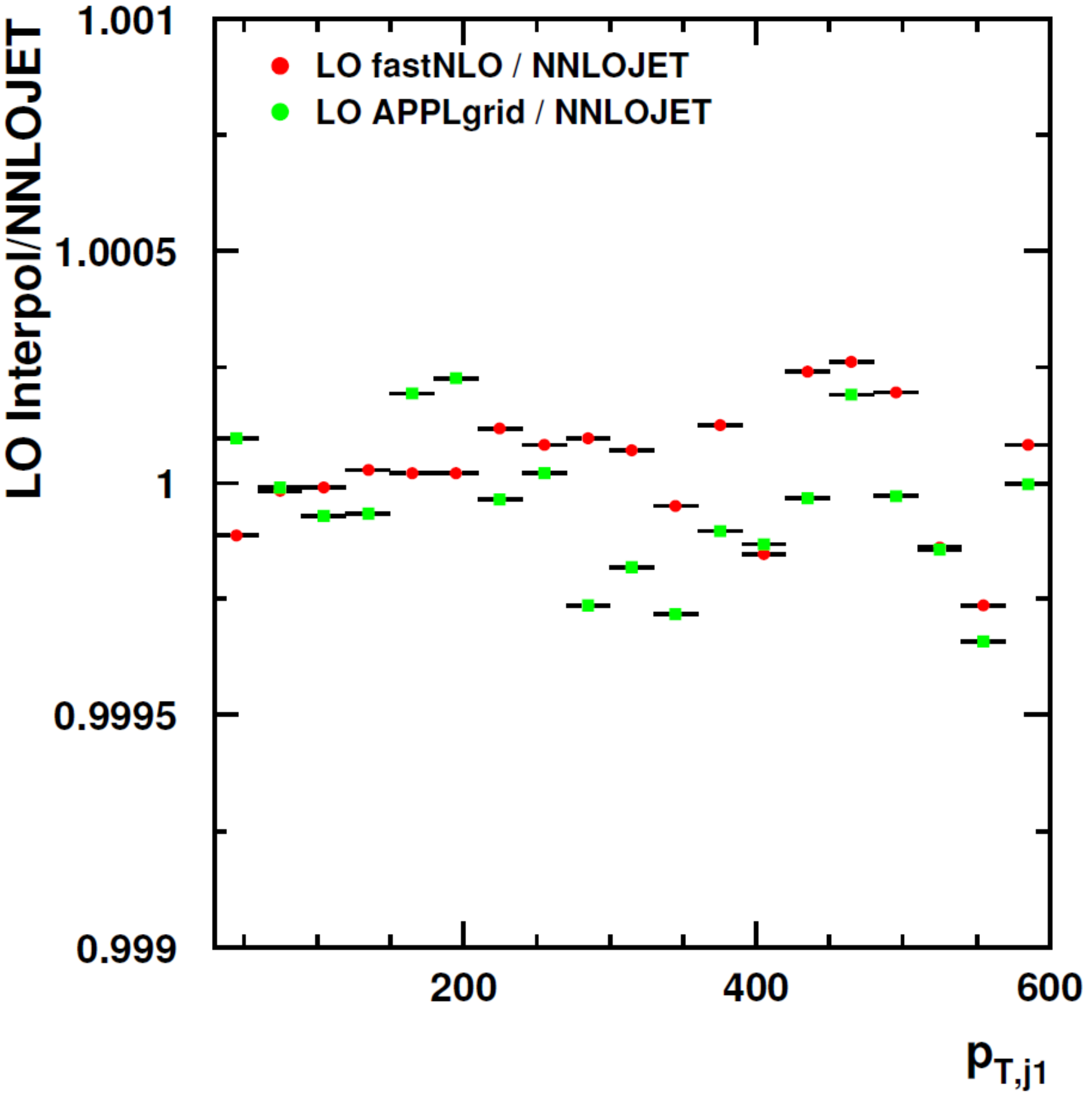
M. Sutton at QCD@LHC 2016

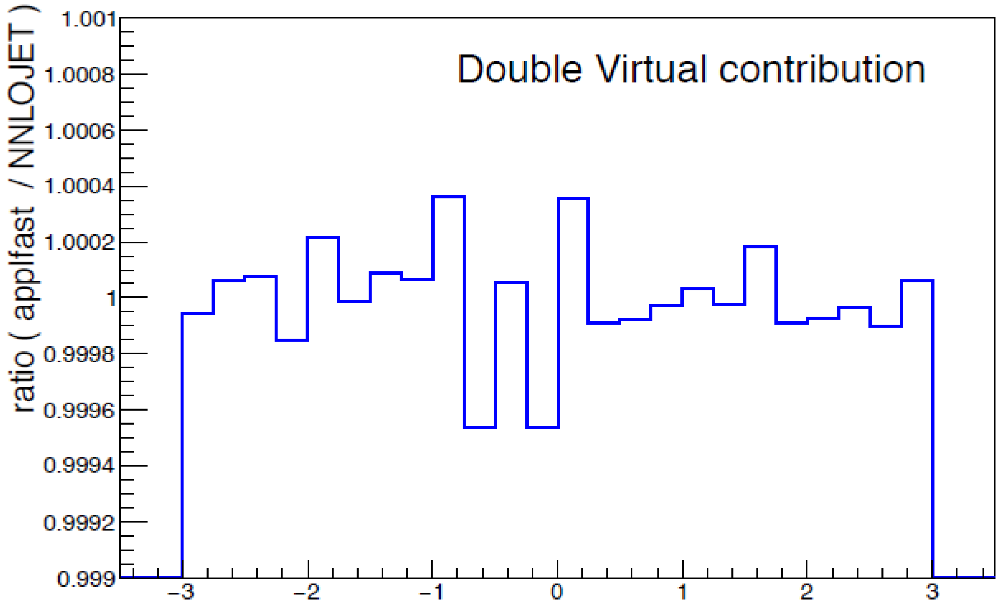
# NNLOJET+APPLfast Workflow

- **1. Preprocessing:** Check of interpolation quality
  - Short test jobs to check interpolation settings (& optimise if necessary)
- **2. NNLOJET Warm-up:** Vegas integration optimisation
  - 1 long (multi-core) job per process
- **3. APPLgrid/fastNLO Warm-up:** Adapt x- and scale-grids to accessed phase space (exact strategy differs between APPLgrid & fastNLO)
  - Only phase space provided from NNLOJET → significant speed-up
- **4. Interpolation grid production:**
  - Thousands of parallel jobs
- **5. Postprocessing:** Statistical evaluation and combination of all produced grids …
  - Work in progress: Combination scripts/programs
- **6. Present final results :-)**

# Step 1: Preprocessing

**Z+jet LO Approximation Test**
**Similar performance at subpermille level**




M. Sutton at QCD@LHC

**Z+jet Test Setup:** $p_{T,j1}$, $\eta_{j1}$, $y_Z$

- $E_{cms}$ = 8 TeV
- $p_{T,jet}$ > 30 GeV
- $|y_{jet}| < 3$
- $|y_{l+,l-}| < 5$
- $80 < M_{l+l-} < 100$ GeV
- $\mu_r = \mu_f = M_Z$ = fixed
  (→ no scale interpolation in this test)

# Step 2: Vegas Integrations

- **NNLOJET Warm-up:**
  - Must be one job per process type
  - Multi-threading possible

| Job Type | # Jobs | Threads / Job | Events / Job | Runtime / Job | Total Runtime |
|---|---|---|---|---|---|
| LO | 1 | 16 | 32 M | 0.35 h | 0.35 h |
| NLO-R | 1 | 16 | 16 M | 1.0 h | 1.0 h |
| NLO-V | 1 | 16 | 16 M | 1.0 h | 1.0 h |
| NNLO-RRa | 1 | 32 | 5 M | 17.5 h | 17.5 h |
| NNLO-RRb | 1 | 32 | 5 M | 20.7 h | 20.7 h |
| NNLO-RV | 1 | 16 | 8 M | 22.4 h | 22.4 h |
| NNLO-VV | 1 | 16 | 8 M | 24.6 h | 24.6 h |
| Total | 7 | - | - | - | 87.6 h |

**Calculated on BwUniCluster at KIT thanks to Baden-Württemberg High Performance Computing (HPC) support**

# Step 3: Phase Space Exploration

- **APPLfast Warm-up:**
  - NNLOJET is run without CPU-time expensive weight calculation
  - At least 1 job per process needed to determine phase space limits individually
  - Grids created and optimised during warm-up (APPLgrid)
  - Grids created in production step from optimised x and Q-scale limits (fastNLO)
  - Warm-up can be parallelised, if necessary (fastNLO)
  - Presented table used for extensive testing; overkill for normal use

In this setup most $x_{min}$ limits from LO runs, 3 from higher-order runs.

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

# Typical Sample Productions

- **NNLOJET Z+jet, no grids:**
  - LO: 100 jobs x (5 min)
  - NLO-V: 100 jobs x (10 min)
  - NLO-R: 100 jobs x (15 min)
  - NNLO-VV: 100 jobs x (1 h)
  - NNLO-RV: 1000 jobs x (10-20h)
  - NNLO-RR: 5000 jobs x (~20h)

**Initial performance penalty of grid production vs. NNLOJET alone reduced to roughly a factor of only ~ 2. Depends in detail on physics process and grid setup.**

**Still gain 100k's of CPU hours for each avoided repetition.**

- **NNLOJET+Grid DIS jets: each job ~8-16h 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)

# Step 4: Mass Production

- **NNLOJET + APPLfast**
  - **Massive parallelised computing on Virtual Machines with 24h lifetime**
  - **Example with fastNLO, APPLgrid example in progress**

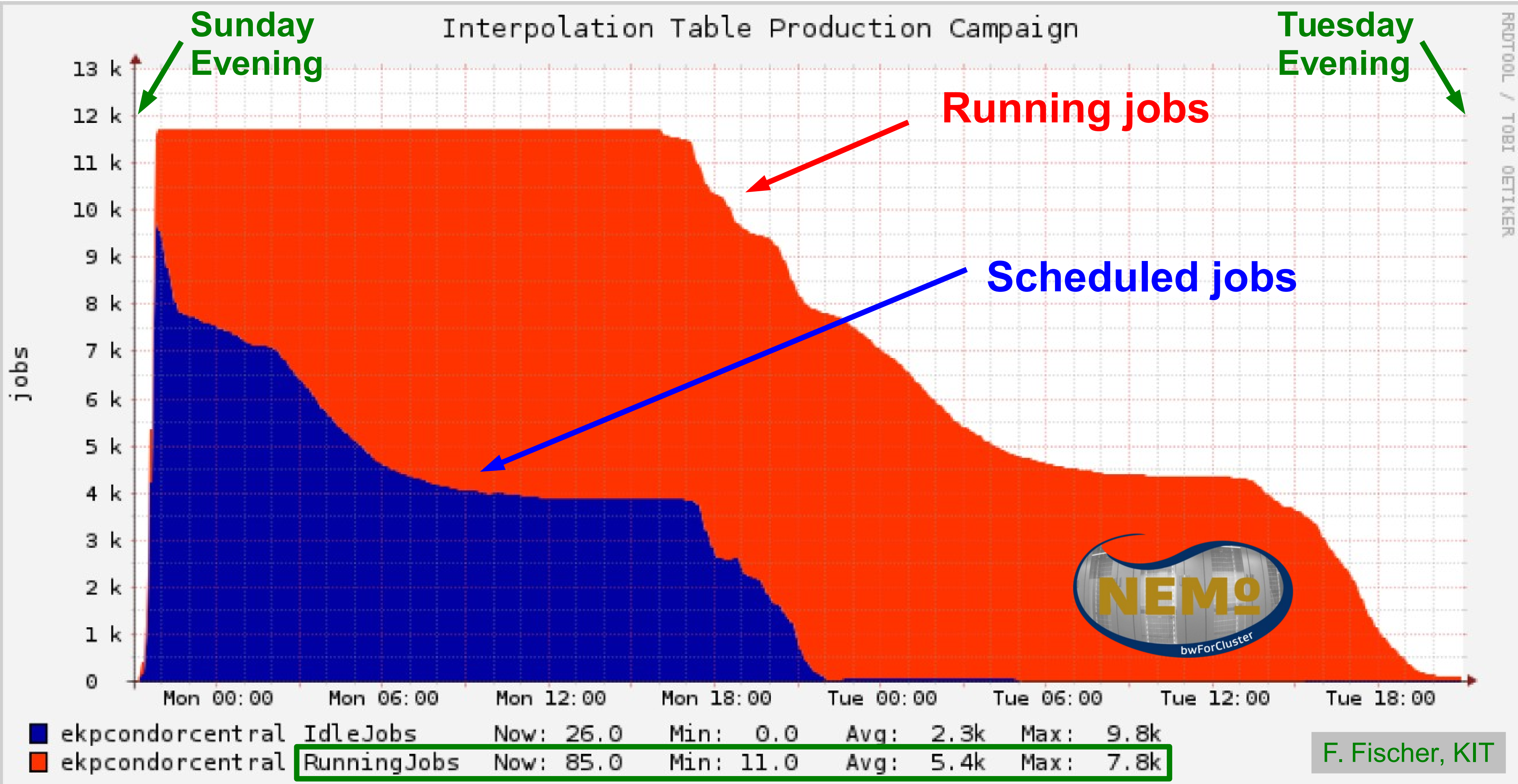
| Job Type | # Jobs | Events / Job | Runtime / Job | # Events | Total Output | Total Runtime |
|---|---|---|---|---|---|---|
| LO | 10 | 140 M | 20.6 h | 1.4 G | 24 MB | 206 h |
| NLO-R | 200 | 6 M | 19.0 h | 1.2 G | 1.3 GB | 3800 h |
| NLO-V | 200 | 5 M | 21.2 h | 1.0 G | 1.2 GB | 4240 h |
| NNLO-RRa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

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

**Calculated on BwForCluster NEMO in Freiburg thanks to Baden-Württemberg High Performance Computing (HPC) support**

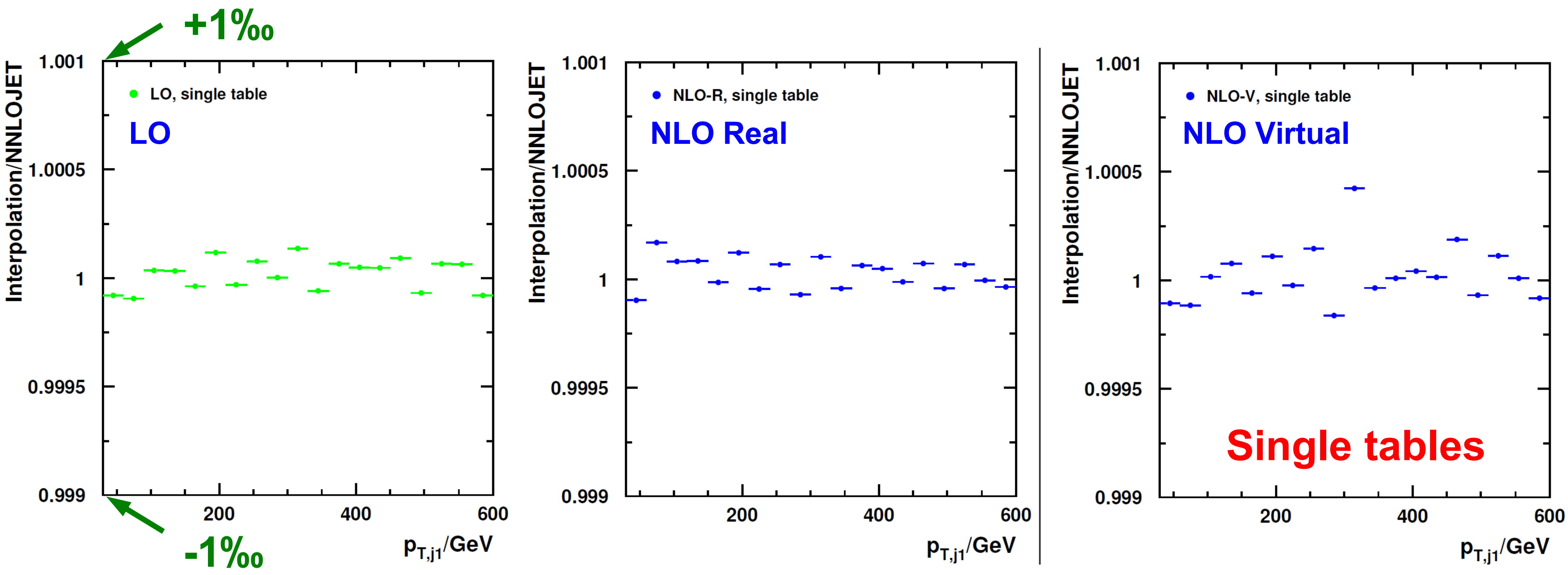
# Production Campaign

**Optimal scenario: Finished in two days with 7800 parallel jobs at maximum!**



# Check LO/NLO Interpolation

## Z+jet approximation test for LO, NLO-R, and NLO-V
## Agreement at subpermille level




**Check using a single interpolation table of the presented production campaign …**

**At this level of precision also LHAPDF grids may become a limiting factor.**

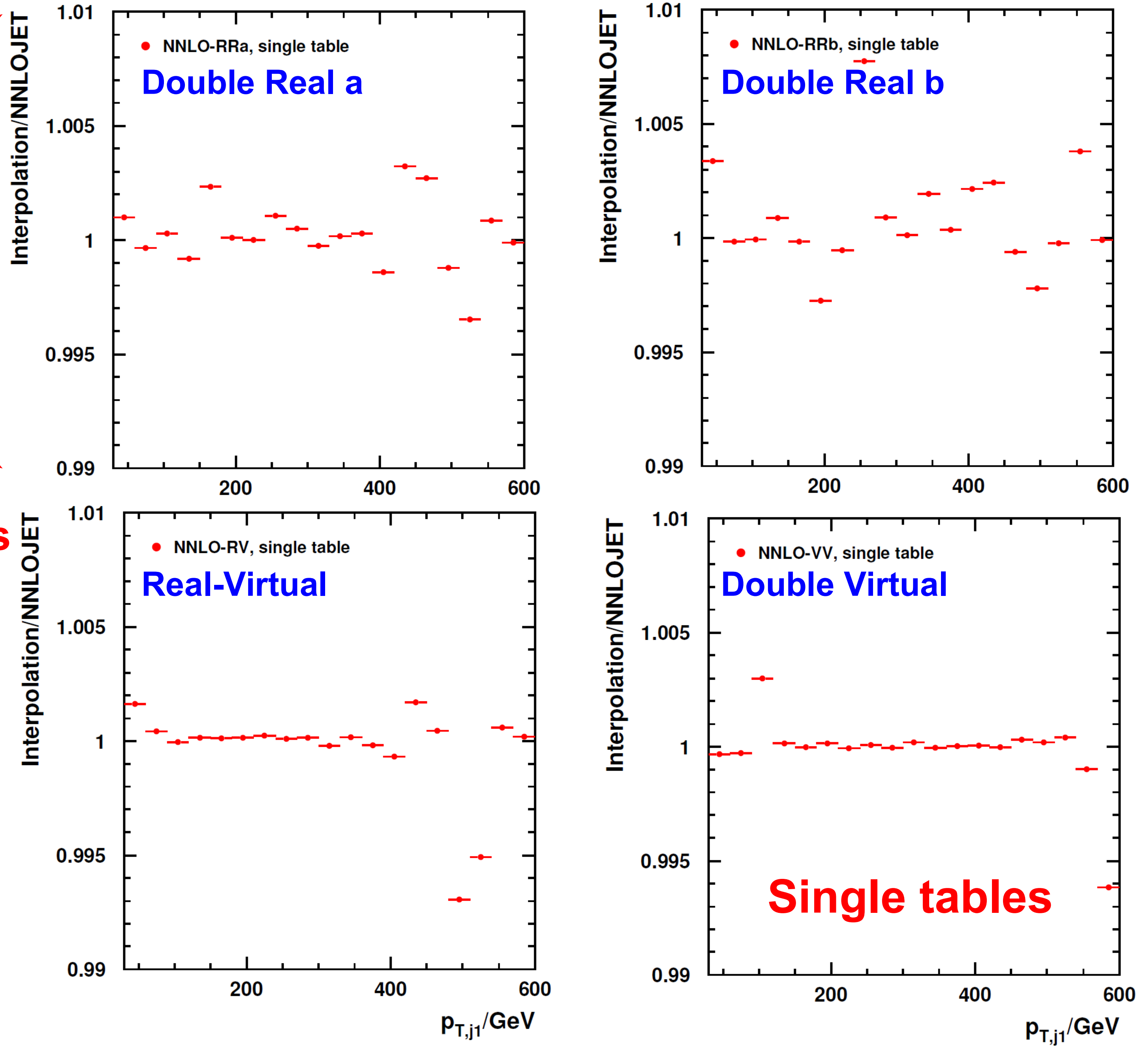
# Check NNLO Interpolation

**+1%**

**Z+jet approximation test for NNLO parts**

**Agreement at subpercent level**

**-1%**

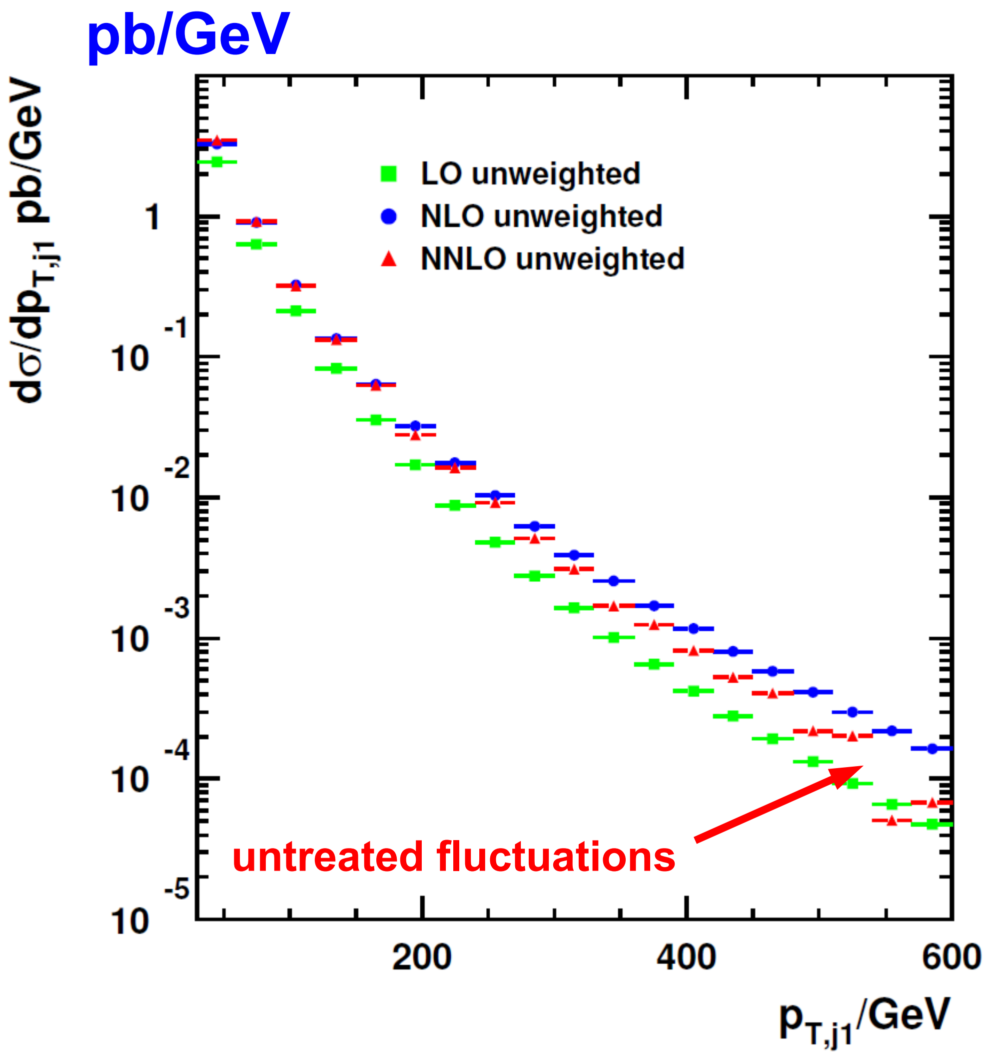


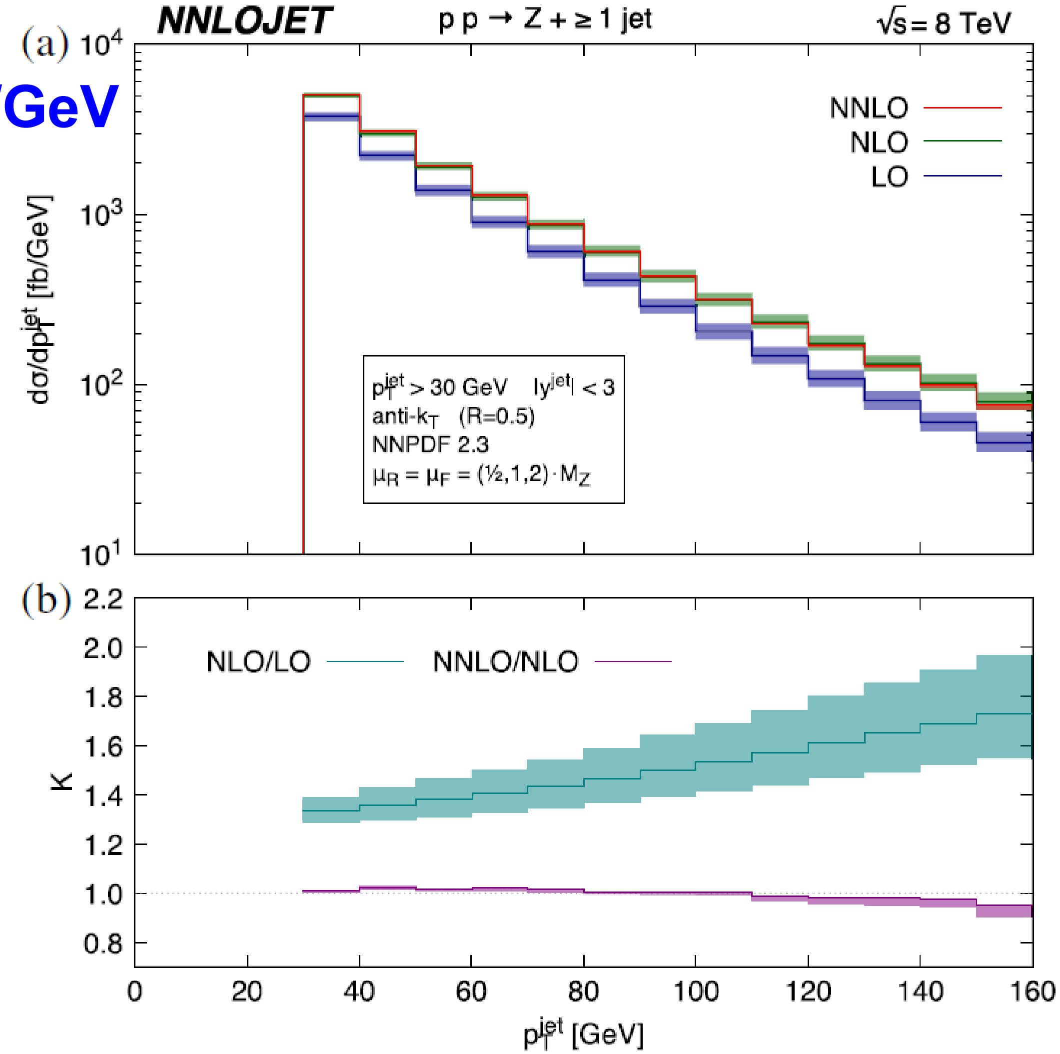

**Some impact of fluctuations visible**
**→ to be dealt with in combination procedure.**

**(a/b indicates a technical phase space separation for this process)**

# Unweighted Combination

**Cross sections from unweighted combination (not final)**
**→ Order of magnitude ok**
**→ Fluctuations still to be dealt with → implement for APPLfast**




Gehrmann-de Ridder et al., PRL, 2016, 117, 022001.

# Outlook

- NNLOJET provides NNLO in common interface for:
  - Z incl., Z+jet, W incl., pp jet+dijets, H incl., H+jet, DIS jet+dijets, e+e- 3jets
  - W+jet almost ready; more to come
- APPLgrid+fastNLO interface (NNLO-Bridge) is working
- Numerous adaptations implemented by all sides
- Large-scale productions tested for Z+jet and DIS jet
- Work in progress: Implementation of final combination procedure for interpolation grids
- Looking forward to many new NNLO interpolation grids in 2017

**We acknowledge support from an IPPP Associateship and Baden-Württemberg HPC support through BwUniCluster and BwForCluster.**

# Backup

# Bridge to Theory Code

**fastNLO example of concept,**
**NNLO bridge works for APPLgrid & fastNLO**

Program Start

(N)NLO Program

MC Integration

```
fastNLOCreate fnlo(„steering.str“);
fnlo.SetOrderOfCalculation(int order);
```

Initialize fastNLO class(es)

```
fnlo.fEvent.SetProcessID(int id);

fnlo.fEvent.SetX1(double x1);
fnlo.fEvent.SetX2(double x2);

fnlo.fEvent.SetWeight(double w);

fnlo.fScenario.SetObservable0(double pt);
fnlo.fScenario.SetObsScale1(double s1);
```

Pass the process specific variables during the 'event loop' to fastNLO

- Order does not matter
- Many other convenient implementations possible

```
fnlo.Fill();
```

Pass all information to fastNLO

```
fnlo.SetNumberOfEvents(double nevents);
fnlo.WriteTable();
```

Set normalization of the MC integration and write table

Program End

(N)NLO Result

fastNLO Table

Minimum implementation:
11 lines of code

**Convenient implementation of fastNLO into any (N)NLO program possible**

# The Interpolation Concept

## Implemented in APPLgrid & fastNLO

APPLgrid, Carli et al., Eur. Phys. J. C, 2010, 66, 503.
fastNLO, Britzger et al., arXiv:0609285, 1208.3641.

## Use interpolation kernel

- Introduce set of n discrete **x-nodes**, $x_i$'s being equidistant in a function f(x)
- Take set of **Eigenfunctions $E_i(x)$** around nodes $x_i$

→ Interpolation kernels

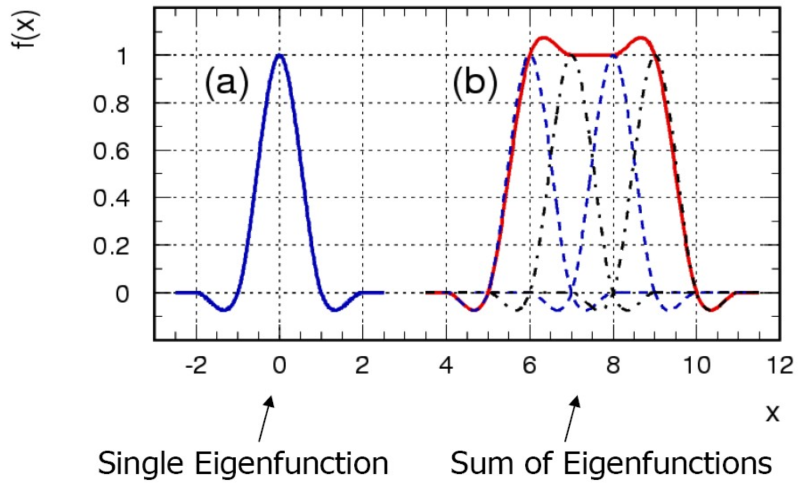
- Actually a rather old idea, see e.g.
  C. Pascaud, F. Zomer (Orsay, LAL), LAL-94-42

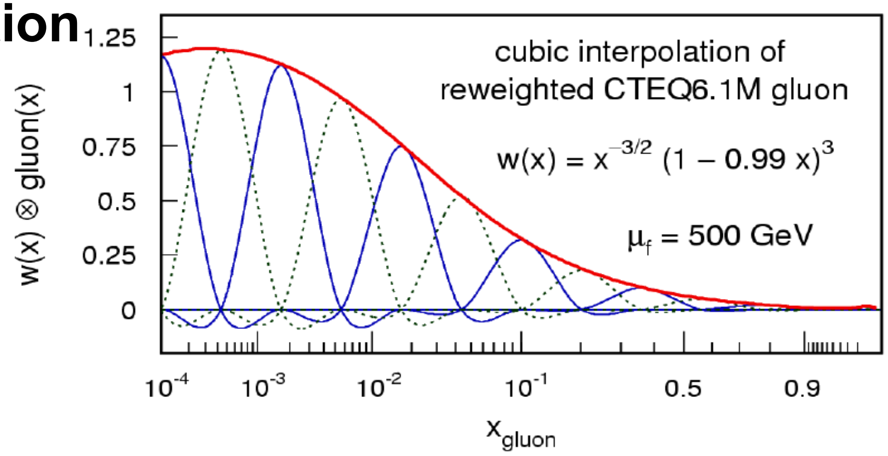
→ Single PDF is replaced by a linear combination of interpolation kernels

$$f_a(x) \cong \sum_i f_a(x_i) \cdot E^{(i)}(x)$$

→ Then the integrals are done only once

→ Afterwards only summation required to change PDF




**Tabulate the convolution of the perturbative coefficients with the interpolation kernel**