Developments in APPLfast

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

Motivation

- Interpretation of experimental data requires reasonably fast theory
- Often: Repeated computation of same cross section:
  - Different PDF sets; PDF uncertainties
  - Variations of renormalisation & factorisation scales μ_R, μ_F
  - Variation of $\alpha_s(M_Z)$
  - SM parameter fits (→ xFitter)
- Jet cross section calculations at NLO were slow → initial reason for interpolation grids
- Nowadays NNLO in general very demanding!
  - Need procedure for fast repeated computations of higher order cross sections
  - Use interpolation grids like from fastNLO or APPLgrid (both are interfaced to xFitter :-))

What is APPLfast?

- Started as common project of NNLOJET, APPLgrid, and fastNLO authors at QCD@LHC in London
- Interface between NNLOJET and fast grid technology - APPLgrid and fastNLO
- Aimed to be the least obtrusive as possible for both ends of the interface
- Intended to be reusable by other theory programs

NOPE, THERE ISN'T ANY MORE TO LIFE! HUNT...GATHER... THAT'S PRETTY MUCH IT!

Bridge function: NNLOJET hooks

- **The interface acts as a bridge between the NNLOJET code and the grid code**
  - **Intention is to be as tries unobtrusive as possible in terms of code disruption - such that neither the NNLOJET nor grid code needs to be recompiled or otherwise changed to exchange information**
  - **Achieved by a novel implementation of unobtrusive hook functions on the NNLOJET side …**

```
/// function pointer hooks - set to 0 when no functions defined and applgrid not linked
void (*appl_initptr)( const int& igrid, const int& id, const std::string& gridname,
                      const int& nbins, const double* bins  ) = 0;

void (*appl_fillptr)( const int& igrid, const int& id, const double& obs, const double* wt ) = 0;

void (*appl_fillrefptr)( const int& igrid, const int& id,
                         const double& obs,
                         const double* wt ) = 0;

void (*appl_termptr)( const int& igrid, const int& id ) = 0;


/// book grids
extern "C" void  book_grid_varbins_( const int& igrid, const int& id,
                                     const int& nbins, const double* bins,
                                     const char* s, size_t len ) {

  if ( appl_initptr ) {
     std::string _s( s, 0, len );
     appl_initptr( igrid, id, _s, nbins, bins );
  }
}
```

  - **If these function pointers are not set, the actual hook functions do nothing**
  - **To run with grid filling, simply need to assign these pointers to point to the actual filling functions**

Bridge operation

- To fill the grids it is necessary to know the **unconvoluted** subprocess weight, without the PDF
- The NNLOJET code is well structured such that extracting this was straight forward
  - A small modification to pass the weight and PDF terms into the histogram filling routine was made (rather than their product)

```
double x1 =  parfrac_.x1;
double x2 =  parfrac_.x2;

double Q2 =  mur2_hook(1);
fill_applgrid( id, x1, x2, Q2, obs, wt );
fill_ref_applgrid( id, obs, wtref );
```

- The actual filling code essentially just a wrapper around such functions for each grid, with scaled momentum fractions …
  - logically the same, although slightly different syntax for each technology (applgrid, fastnlo)

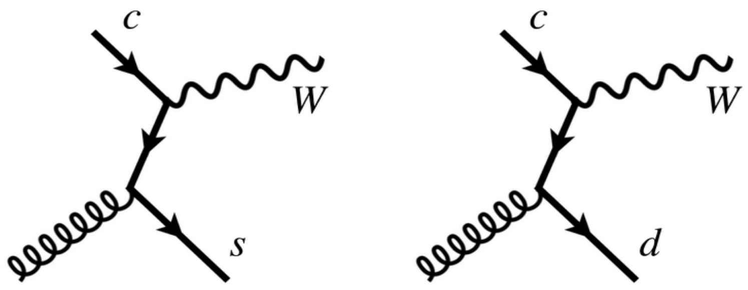
```
int ix = gridix_.igx; /// get this from our new, added common block

double _x1 = x1;
double _x2 = x2;

if      ( ix==2 ) _x2 = x2/xregions_.xreg2;
else if ( ix==3 ) _x1 = x1/xregions_.xreg1;
else if ( ix==4 ) {
  _x1 = x1/xregions_.xreg1;
  _x2 = x2/xregions_.xreg2;
}

if ( jacobian_.jflag ) jac *= 1/((1-x1)*(1-x2));  /// *not* 1/((1-_x1)(1-_x2))

if      ( ix==2 ) jac *= 1./xregions_.xreg2;
else if ( ix==3 ) jac *= 1./xregions_.xreg1;
else if ( ix==4 ) jac *= 1./(xregions_.xreg1*xregions_.xreg2);


if ( !getunitphase_() && usecache ) {
  fillcache( _grid, pdf->size(), obs, order, iprocess, _x1, _x2, Q2,  wgt[iprocess]*asfactor*jac );
}
else {
  weight[iprocess] = wgt[iprocess]*asfactor*jac;

  if ( first_run ) _grid->fill_phasespace(  _x1, _x2, Q2, obs,  &weight[0], order );
  else             _grid->fill_grid(        _x1, _x2, Q2, obs,  &weight[0], order );
}
```

Bridge code caching

- On a grid fill, in principle each weight is spread over
  - Interpolation Order(x1) × Interpolation Order(x2) × Interpolation Order(Q2) grid nodes
- Requires calculation of IntOrd(x1) + IntOrd(x2) + IntOrd(Q2) interpolating coefficients
- Each grid fill can trigger to regenerate all these interpolating coefficients → it can be useful to cache all the weights before filling
  - Coefficients depend not just on x1, x2, Q2 etc, but also on the grid nodes, which in turn depend on the observable bin → coefficients will be different for different observables
- NNLOJET code loops through different >> 100 internal subprocesses often with common x1, x2, Q2 points for some small range of shots
  - A caching mechanism was implemented for each grid
    - Check whether any of the x1, x2, Q2, observable and order have changed for each grid
    - If unchanged, do not fill grid - keep a running sum of each weight contribution
    - Only actually fill the grid if any of the kinematics have changed
- In principle this caching could me migrated into the grids themselves ...

Parton "luminosity subprocesses"

- Potentially can have - neglecting top - 121 different parton-parton luminosities for any calculation
- Can even be larger, if hard subprocess requires multiple luminosities with the same input
  - e.g. NNLO contribution to W production if we want to retain ability to vary (or fit) the CKM matrix terms



- Have implemented available subprocesses using a class encoding the parton-parton contributions, and allows duplicated initial state contributions
  - We make use of this functionality also to have allow the different NNLOJET subprocess contributions to map to a reduced number of actual luminosities that we store …
- As an example ,for Z+jet production the LO contributions from NNLOJET are …

0	1	$(d, g) + (s, g) + (b, g)$
1	2	$(u, g) + (c, g)$
2	3	$(d, \bar{d}) + (s, \bar{s}) + (b, \bar{b})$
3	4	$(u, \bar{u}) + (c, \bar{c})$
4	5	$(g, d) + (g, s) + (g, b)$
5	6	$(g, u) + (g, c)$
6	7	$(\bar{d}, d) + (\bar{s}, s) + (\bar{b}, b)$
7	8	$(\bar{u}, u) + (\bar{c}, c)$
8	9	$(g, \bar{d}) + (g, \bar{s}) + (g, \bar{b})$
9	10	$(g, \bar{u}) + (g, \bar{c})$
10	11	$(\bar{d}, g) + (\bar{s}, g) + (\bar{b}, g)$
11	12	$(\bar{u}, g) + (\bar{c}, g)$

```
[ ( 1) :        (1, 0)  (3, 0)  (5, 0) ]
[ ( 2) :        (2, 0)  (4, 0) ]
[ ( 3) :        (1, -1) (3, -3) (5, -5) ]
[ ( 4) :        (2, -2) (4, -4) ]
[ ( 5) :        (0, 1)  (0, 3)  (0, 5) ]
[ ( 6) :        (0, 2)  (0, 4) ]
[ ( 7) :        (-1, 1) (-3, 3) (-5, 5) ]
[ ( 8) :        (-2, 2) (-4, 4) ]
[ ( 9) :        (0, -1) (0, -3) (0, -5) ]
[ ( 10) :       (0, -2) (0, -4) ]
[ ( 11) :       (-1, 0) (-3, 0) (-5, 0) ]
[ ( 12) :       (-2, 0) (-4, 0) ]
```

- Which has 12 contributions - no remapping

Z + jets at NNLO

- **The same 33 input parton luminosities as in the NLO case, however, many more (~ 800) individual internal processes ...**

0	163 177 191 205 206 245 246 285 301 317 347 377 391 405 435 449 450 489 490 529 530 569 593 617 647 677 707 737 767 899 905 906 935 941 942	$(d,\bar{d}) + (s,\bar{s}) + (b,\bar{b})$
1	164 178 192 207 208 247 248 286 302 318 348 378 392 406 436 451 452 491 492 531 532 570 594 618 648 678 708 738 768 900 907 908 936 943 944	$(u,\bar{u}) + (c,\bar{c})$
2	165 179 193 209 210 249 250 287 303 319 349 379 393 407 437 453 454 493 494 533 534 571 595 619 649 679 709 739 769 909 917 918 945 953 954	$(\bar{d},d) + (\bar{s},s) + (\bar{b},b)$
3	166 180 194 211 212 251 252 288 304 320 350 380 394 408 438 455 456 495 496 535 536 572 596 620 650 680 710 740 770 910 919 920 946 955 956	$(\bar{u},u) + (\bar{c},c)$
4	167 181 195 229 230 269 270 289 305 321 351 381 395 409 439 481 482 521 522 561 562 585 586 609 610 621 651 681 711 741 771 817 818 861 862 901 902 937 938	$(d,g) + (s,g) + (b,g)$
5	168 182 196 231 232 271 272 290 306 322 352 382 396 410 440 483 484 523 524 563 564 587 588 611 612 622 652 682 712 742 772 819 820 863 864 903 904 939 940	$(u,g) + (c,g)$
6	169 183 197 233 234 273 274 293 309 323 353 383 397 411 441 473 474 513 514 553 554 577 578 601 602 623 653 683 713 743 773 801 802 845 846 889 890 925 926	$(g,d) + (g,s) + (g,b)$
7	170 184 198 235 236 275 276 294 310 324 354 384 398 412 442 475 476 515 516 555 556 579 580 603 604 624 654 684 714 744 774 803 804 847 848 891 892 927 928	$(g,u) + (g,c)$
8	171 172 185 186 199 200 325 326 355 356 385 386 399 400 413 414 443 444 629 630 659 660 689 690 719 720 749 750 779 780 797 798 799 800 841 842 843 844 885 886 887 888 921 922 923 924	(g,g)
9	173 187 201 237 238 277 278 295 311 327 357 387 401 415 445 477 478 517 518 557 558 581 582 605 606 625 655 685 715 745 775 805 806 849 850 893 894 929 930	$(g,\bar{d}) + (g,\bar{s}) + (g,\bar{b})$
10	174 188 202 239 240 279 280 296 312 328 358 388 402 416 446 479 480 519 520 559 560 583 584 607 608 626 656 686 716 746 776 807 808 851 852 895 896 931 932	$(g,\bar{u}) + (g,\bar{c})$
11	175 189 203 241 242 281 282 297 313 329 359 389 403 417 447 485 486 525 526 565 566 589 590 613 614 627 657 687 717 747 777 833 834 877 878 913 914 949 950	$(\bar{d},g) + (\bar{s},g) + (\bar{b},g)$
12	176 190 204 243 244 283 284 298 314 330 360 390 404 418 448 487 488 527 528 567 568 591 592 615 616 628 658 688 718 748 778 835 836 879 880 915 916 951 952	$(\bar{u},g) + (\bar{c},g)$
13	213 253 335 365 423 457 497 537 635 665 695 725 755 785 813 821 822 857 865 866	$(d,\bar{d}) + (d,\bar{s}) + (d,\bar{b}) + (s,\bar{d}) + (s,\bar{s}) + (s,\bar{b}) + (b,\bar{d}) + (b,\bar{s}) + (b,\bar{b})$
14	214 254 336 366 424 458 498 538 636 666 696 726 756 786 814 858	$(d,\bar{u}) + (d,\bar{c}) + (s,\bar{u}) + (s,\bar{c}) + (b,\bar{u}) + (b,\bar{c})$
15	215 255 337 367 425 459 499 539 637 667 697 727 757 787 815 859	$(u,\bar{d}) + (u,\bar{s}) + (u,\bar{b}) + (c,\bar{d}) + (c,\bar{s}) + (c,\bar{b})$
16	216 256 338 368 426 460 500 540 638 668 698 728 758 788 816 823 824 860 867 868	$(u,\bar{u}) + (u,\bar{c}) + (c,\bar{u}) + (c,\bar{c})$
17	217 257 331 361 419 461 501 541 631 661 691 721 751 781 809 853	$(d,d) + (d,s) + (d,b) + (s,d) + (s,s) + (s,b) + (b,d) + (b,s) + (b,b)$
18	218 258 332 362 420 462 502 542 632 662 692 722 752 782 810 854	$(d,u) + (d,c) + (s,u) + (s,c) + (b,u) + (b,c)$
19	219 259 333 363 421 463 503 543 633 663 693 723 753 783 811 855	$(u,d) + (u,s) + (u,b) + (c,d) + (c,s) + (c,b)$
20	220 260 334 364 422 464 504 544 634 664 694 724 754 784 812 856	$(u,u) + (u,c) + (c,u) + (c,c)$
21	221 261 343 373 431 465 505 545 643 673 703 733 763 793 829 873	$(\bar{d},\bar{d}) + (\bar{d},\bar{s}) + (\bar{d},\bar{b}) + (\bar{s},\bar{d}) + (\bar{s},\bar{s}) + (\bar{s},\bar{b}) + (\bar{b},\bar{d}) + (\bar{b},\bar{s}) + (\bar{b},\bar{b})$
22	222 262 344 374 432 466 506 546 644 674 704 734 764 794 830 874	$(\bar{d},\bar{u}) + (\bar{d},\bar{c}) + (\bar{s},\bar{u}) + (\bar{s},\bar{c}) + (\bar{b},\bar{u}) + (\bar{b},\bar{c})$
23	223 263 345 375 433 467 507 547 645 675 705 735 765 795 831 875	$(\bar{u},\bar{d}) + (\bar{u},\bar{s}) + (\bar{u},\bar{b}) + (\bar{c},\bar{d}) + (\bar{c},\bar{s}) + (\bar{c},\bar{b})$
24	224 264 346 376 434 468 508 548 646 676 706 736 766 796 832 876	$(\bar{u},\bar{u}) + (\bar{u},\bar{c}) + (\bar{c},\bar{u}) + (\bar{c},\bar{c})$
25	225 265 339 369 427 469 509 549 639 669 699 729 759 789 825 837 838 869 881 882	$(\bar{d},d) + (\bar{d},s) + (\bar{d},b) + (\bar{s},d) + (\bar{s},s) + (\bar{s},b) + (\bar{b},d) + (\bar{b},s) + (\bar{b},b)$
26	226 266 340 370 428 470 510 550 640 670 700 730 760 790 826 870	$(\bar{d},u) + (\bar{d},c) + (\bar{s},u) + (\bar{s},c) + (\bar{b},u) + (\bar{b},c)$
27	227 267 341 371 429 471 511 551 641 671 701 731 761 791 827 871	$(\bar{u},d) + (\bar{u},s) + (\bar{u},b) + (\bar{c},d) + (\bar{c},s) + (\bar{c},b)$
28	228 268 342 372 430 472 512 552 642 672 702 732 762 792 828 839 840 872 883 884	$(\bar{u},u) + (\bar{u},c) + (\bar{c},u) + (\bar{c},c)$
29	291 307 573 597 897 933	$(d,d) + (s,s) + (b,b)$
30	292 308 574 598 898 934	$(u,u) + (c,c)$
31	299 315 575 599 911 947	$(\bar{d},\bar{d}) + (\bar{s},\bar{s}) + (\bar{b},\bar{b})$
32	300 316 576 600 912 948	$(\bar{u},\bar{u}) + (\bar{c},\bar{c})$

Jet production at NNLO

- **Double virtual - 93 internal processes - but only 11 distinct parton luminosities**

0	270 279 288 301 312 323 336 347 358	(g, g)
1	271 280 289 299 310 321 337 348 359	$(g, d) + (g, u) + (g, s) + (g, c) + (g, b)$
2	272 281 290 302 313 324 338 349 360	$(g, \bar{d}) + (g, \bar{u}) + (g, \bar{s}) + (g, \bar{c}) + (g, \bar{b})$
3	273 282 291 304 315 326 331 342 353	$(d, d) + (d, u) + (d, s) + (d, c) + (d, b) + (u, d) + (u, u) + (u, s) + (u, c) + (u, b) + (s, d) + (s, u) + (s, s) + (s, c) + (s, b) + (c, d) + (c, u) + (c, s) + (c, c) + (c, b) + (b, d) + (b, u) + (b, s) + (b, c) + (b, b)$
4	274 283 292 305 316 327 330 341 352	$(d, \bar{d}) + (d, \bar{u}) + (d, \bar{s}) + (d, \bar{c}) + (d, \bar{b}) + (u, \bar{d}) + (u, \bar{u}) + (u, \bar{s}) + (u, \bar{c}) + (u, \bar{b}) + (s, \bar{d}) + (s, \bar{u}) + (s, \bar{s}) + (s, \bar{c}) + (s, \bar{b}) + (c, \bar{d}) + (c, \bar{u}) + (c, \bar{s}) + (c, \bar{c}) + (c, \bar{b}) + (b, \bar{d}) + (b, \bar{u}) + (b, \bar{s}) + (b, \bar{c}) + (b, \bar{b})$
5	275 284 293 297 308 319 339 350 361	$(d, g) + (u, g) + (s, g) + (c, g) + (b, g)$
6	276 285 294 306 317 328 335 346 357	$(\bar{d}, d) + (\bar{d}, u) + (\bar{d}, s) + (\bar{d}, c) + (\bar{d}, b) + (\bar{u}, d) + (\bar{u}, u) + (\bar{u}, s) + (\bar{u}, c) + (\bar{u}, b) + (\bar{s}, d) + (\bar{s}, u) + (\bar{s}, s) + (\bar{s}, c) + (\bar{s}, b) + (\bar{c}, d) + (\bar{c}, u) + (\bar{c}, s) + (\bar{c}, c) + (\bar{c}, b) + (\bar{b}, d) + (\bar{b}, u) + (\bar{b}, s) + (\bar{b}, c) + (\bar{b}, b)$
7	277 286 295 307 318 329 334 345 356	$(\bar{d}, \bar{d}) + (\bar{d}, \bar{u}) + (\bar{d}, \bar{s}) + (\bar{d}, \bar{c}) + (\bar{d}, \bar{b}) + (\bar{u}, \bar{d}) + (\bar{u}, \bar{u}) + (\bar{u}, \bar{s}) + (\bar{u}, \bar{c}) + (\bar{u}, \bar{b}) + (\bar{s}, \bar{d}) + (\bar{s}, \bar{u}) + (\bar{s}, \bar{s}) + (\bar{s}, \bar{c}) + (\bar{s}, \bar{b}) + (\bar{c}, \bar{d}) + (\bar{c}, \bar{u}) + (\bar{c}, \bar{s}) + (\bar{c}, \bar{c}) + (\bar{c}, \bar{b}) + (\bar{b}, \bar{d}) + (\bar{b}, \bar{u}) + (\bar{b}, \bar{s}) + (\bar{b}, \bar{c}) + (\bar{b}, \bar{b})$
8	278 287 296 303 314 325 340 351 362	$(\bar{d}, g) + (\bar{u}, g) + (\bar{s}, g) + (\bar{c}, g) + (\bar{b}, g)$
9	298 309 320 332 343 354	$(d, \bar{d}) + (u, \bar{u}) + (s, \bar{s}) + (c, \bar{c}) + (b, \bar{b})$
10	300 311 322 333 344 355	$(\bar{d}, d) + (\bar{u}, u) + (\bar{s}, s) + (\bar{c}, c) + (\bar{b}, b)$

- **Real-virtual - 54 internal processes**

0	216 221 230 241 258 269	(g, g)
1	217 222 228 239 253 264	$(g, d) + (g, u) + (g, s) + (g, c) + (g, b)$
2	218 223 231 242 257 268	$(g, \bar{d}) + (g, \bar{u}) + (g, \bar{s}) + (g, \bar{c}) + (g, \bar{b})$
3	219 224 226 237 251 262	$(d, g) + (u, g) + (s, g) + (c, g) + (b, g)$
4	220 225 232 243 256 267	$(\bar{d}, g) + (\bar{u}, g) + (\bar{s}, g) + (\bar{c}, g) + (\bar{b}, g)$
5	227 238 248 259	$(d, \bar{d}) + (u, \bar{u}) + (s, \bar{s}) + (c, \bar{c}) + (b, \bar{b})$
6	229 240 252 263	$(\bar{d}, d) + (\bar{u}, u) + (\bar{s}, s) + (\bar{c}, c) + (\bar{b}, b)$
7	233 244 249 260	$(d, d) + (d, u) + (d, s) + (d, c) + (d, b) + (u, d) + (u, u) + (u, s) + (u, c) + (u, b) + (s, d) + (s, u) + (s, s) + (s, c) + (s, b) + (c, d) + (c, u) + (c, s) + (c, c) + (c, b) + (b, d) + (b, u) + (b, s) + (b, c) + (b, b)$
8	234 245 250 261	$(d, \bar{d}) + (d, \bar{u}) + (d, \bar{s}) + (d, \bar{c}) + (d, \bar{b}) + (u, \bar{d}) + (u, \bar{u}) + (u, \bar{s}) + (u, \bar{c}) + (u, \bar{b}) + (s, \bar{d}) + (s, \bar{u}) + (s, \bar{s}) + (s, \bar{c}) + (s, \bar{b}) + (c, \bar{d}) + (c, \bar{u}) + (c, \bar{s}) + (c, \bar{c}) + (c, \bar{b}) + (b, \bar{d}) + (b, \bar{u}) + (b, \bar{s}) + (b, \bar{c}) + (b, \bar{b})$
9	235 246 254 265	$(\bar{d}, d) + (\bar{d}, u) + (\bar{d}, s) + (\bar{d}, c) + (\bar{d}, b) + (\bar{u}, d) + (\bar{u}, u) + (\bar{u}, s) + (\bar{u}, c) + (\bar{u}, b) + (\bar{s}, d) + (\bar{s}, u) + (\bar{s}, s) + (\bar{s}, c) + (\bar{s}, b) + (\bar{c}, d) + (\bar{c}, u) + (\bar{c}, s) + (\bar{c}, c) + (\bar{c}, b) + (\bar{b}, d) + (\bar{b}, u) + (\bar{b}, s) + (\bar{b}, c) + (\bar{b}, b)$
10	236 247 255 266	$(\bar{d}, \bar{d}) + (\bar{d}, \bar{u}) + (\bar{d}, \bar{s}) + (\bar{d}, \bar{c}) + (\bar{d}, \bar{b}) + (\bar{u}, \bar{d}) + (\bar{u}, \bar{u}) + (\bar{u}, \bar{s}) + (\bar{u}, \bar{c}) + (\bar{u}, \bar{b}) + (\bar{s}, \bar{d}) + (\bar{s}, \bar{u}) + (\bar{s}, \bar{s}) + (\bar{s}, \bar{c}) + (\bar{s}, \bar{b}) + (\bar{c}, \bar{d}) + (\bar{c}, \bar{u}) + (\bar{c}, \bar{s}) + (\bar{c}, \bar{c}) + (\bar{c}, \bar{b}) + (\bar{b}, \bar{d}) + (\bar{b}, \bar{u}) + (\bar{b}, \bar{s}) + (\bar{b}, \bar{c}) + (\bar{b}, \bar{b})$

- **Double real - 25 internal processes**

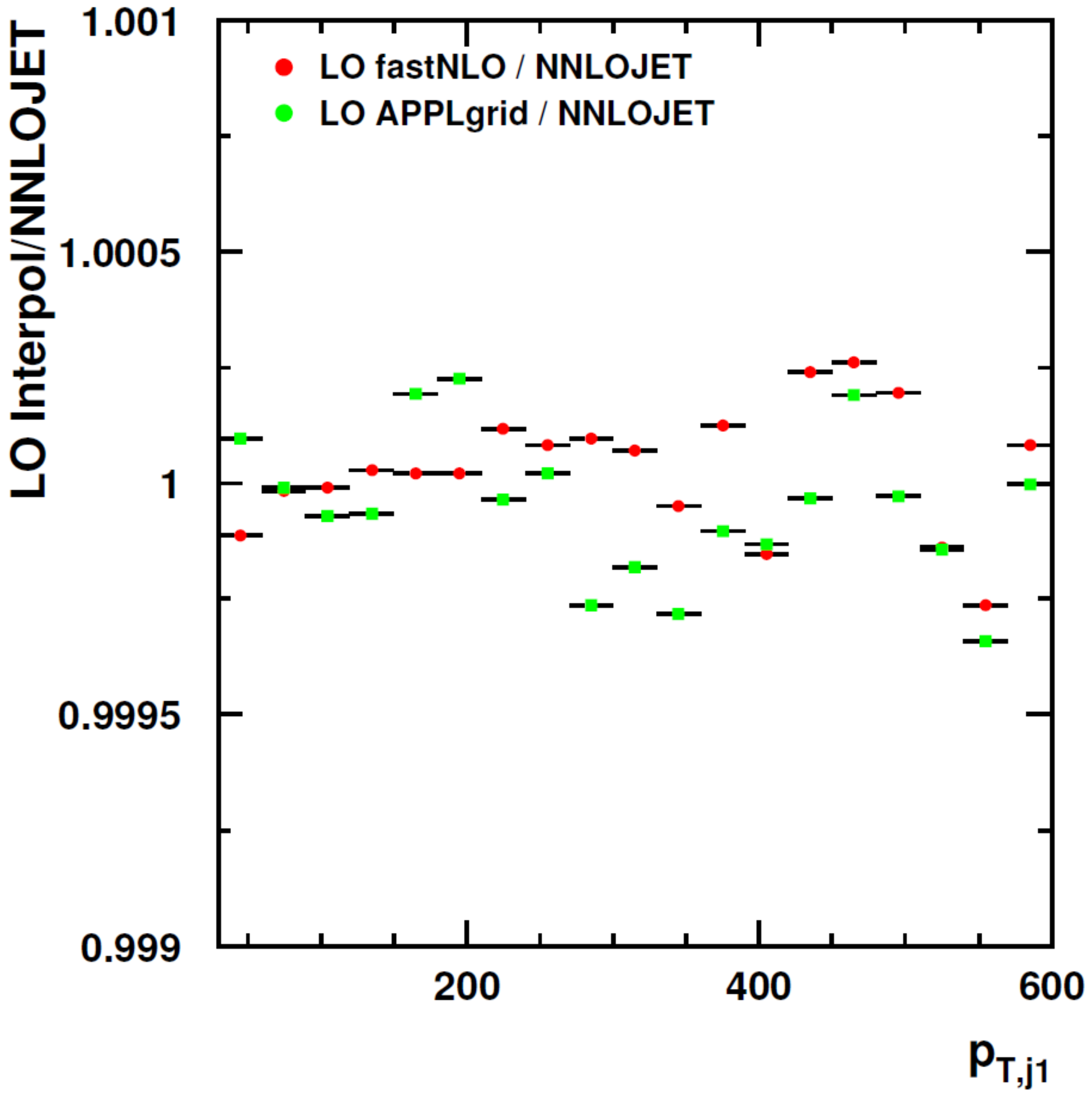
0	191 196 209	(g, g)
1	192 202	$(d, g) + (u, g) + (s, g) + (c, g) + (b, g)$
2	193 199 210	$(d, \bar{d}) + (u, \bar{u}) + (s, \bar{s}) + (c, \bar{c}) + (b, \bar{b})$
3	194 204	$(g, d) + (g, u) + (g, s) + (g, c) + (g, b)$
4	195 203 213	$(\bar{d}, d) + (\bar{u}, u) + (\bar{s}, s) + (\bar{c}, c) + (\bar{b}, b)$
5	197 208	$(g, \bar{d}) + (g, \bar{u}) + (g, \bar{s}) + (g, \bar{c}) + (g, \bar{b})$
6	198 207	$(\bar{d}, g) + (\bar{u}, g) + (\bar{s}, g) + (\bar{c}, g) + (\bar{b}, g)$
7	200 211	$(d, d) + (d, u) + (d, s) + (d, c) + (d, b) + (u, d) + (u, u) + (u, s) + (u, c) + (u, b) + (s, d) + (s, u) + (s, s) + (s, c) + (s, b) + (c, d) + (c, u) + (c, s) + (c, c) + (c, b) + (b, d) + (b, u) + (b, s) + (b, c) + (b, b)$
8	201 212	$(d, \bar{d}) + (d, \bar{u}) + (d, \bar{s}) + (d, \bar{c}) + (d, \bar{b}) + (u, \bar{d}) + (u, \bar{u}) + (u, \bar{s}) + (u, \bar{c}) + (u, \bar{b}) + (s, \bar{d}) + (s, \bar{u}) + (s, \bar{s}) + (s, \bar{c}) + (s, \bar{b}) + (c, \bar{d}) + (c, \bar{u}) + (c, \bar{s}) + (c, \bar{c}) + (c, \bar{b}) + (b, \bar{d}) + (b, \bar{u}) + (b, \bar{s}) + (b, \bar{c}) + (b, \bar{b})$
9	205 214	$(\bar{d}, d) + (\bar{d}, u) + (\bar{d}, s) + (\bar{d}, c) + (\bar{d}, b) + (\bar{u}, d) + (\bar{u}, u) + (\bar{u}, s) + (\bar{u}, c) + (\bar{u}, b) + (\bar{s}, d) + (\bar{s}, u) + (\bar{s}, s) + (\bar{s}, c) + (\bar{s}, b) + (\bar{c}, d) + (\bar{c}, u) + (\bar{c}, s) + (\bar{c}, c) + (\bar{c}, b) + (\bar{b}, d) + (\bar{b}, u) + (\bar{b}, s) + (\bar{b}, c) + (\bar{b}, b)$
10	[illegible] 215	$(\bar{d}, \bar{d}) + (\bar{d}, \bar{u}) + (\bar{d}, \bar{s}) + (\bar{d}, \bar{c}) + (\bar{d}, \bar{b}) + (\bar{u}, \bar{d}) + (\bar{u}, \bar{u}) + (\bar{u}, \bar{s}) + (\bar{u}, \bar{c}) + (\bar{u}, \bar{b}) + (\bar{s}, \bar{d}) + (\bar{s}, \bar{u}) + (\bar{s}, \bar{s}) + (\bar{s}, \bar{c}) + (\bar{s}, \bar{b}) + (\bar{c}, \bar{d}) + (\bar{c}, \bar{u}) + (\bar{c}, \bar{s}) + (\bar{c}, \bar{c}) +$ [illegible]

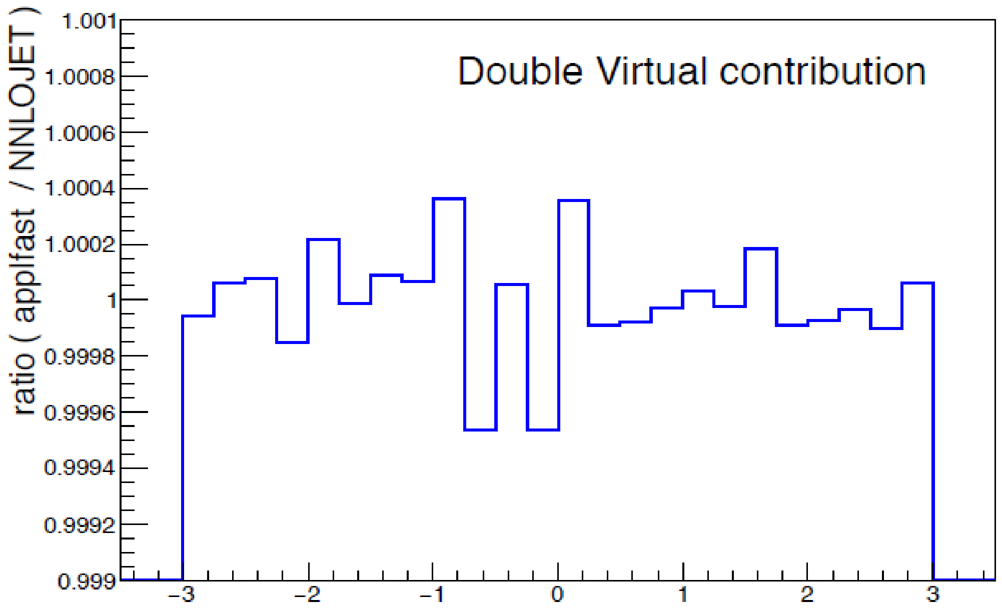
NNLOJET+APPLfast Workflow

- **1. Preprocessing:** Check of interpolation quality
  - Short test jobs to check interpolation settings (& optimise if necessary) O(10 h)
- **2. NNLOJET Warm-up:** Vegas integration optimisation
  - 1 long (multi-core) job per process O(100 h)
- **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 O(100 h)
- **4. Interpolation grid production:**
  - Thousands of parallel jobs O(250 kh)
- **5. Postprocessing:** Statistical evaluation and combination of all produced grids …
  - Job to combine all grids and estimate statistical uncertainty O(100 h)
- **6. Validate, validate, and validate** O(? h)
- **7. Present final results :-)** 30 min :-)

Step 1: Preprocessing

Z+jet LO Approximation Test
Similar performance at subpermille level





M. Sutton at QCD@LHC

Z+jet Test Setup: $p_{T,j1}$, η_{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

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

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

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

Step 4: Mass Production

NNLOJET + APPLfast

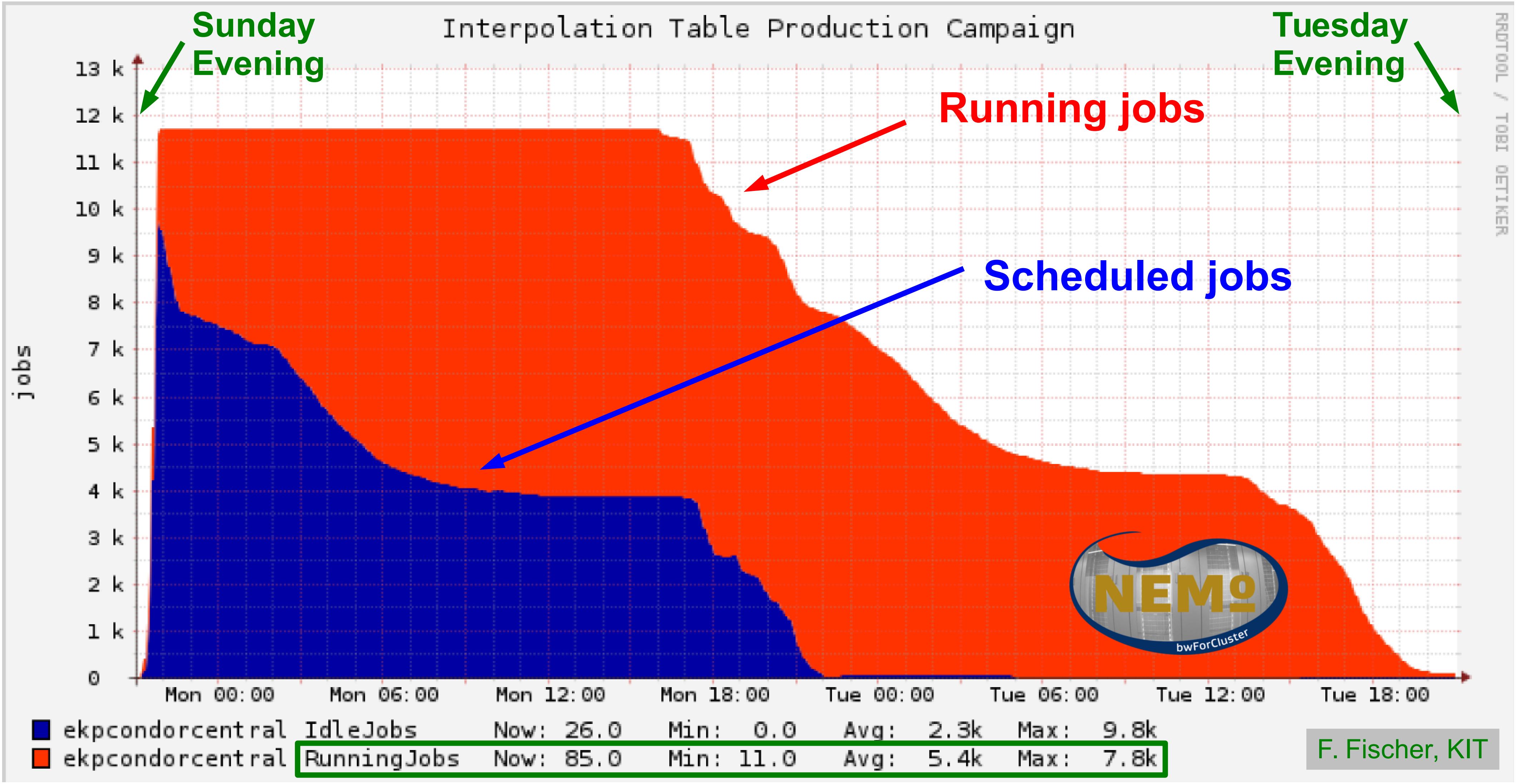
- Massive parallelised computing on Virtual Machines with 24h lifetime
- Example with fastNLO, APPLgrid example in progress

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.

Production Campaign

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



```
■ ekpcondorcentral IdleJobs     Now: 26.0   Min:  0.0   Avg: 2.3k   Max: 9.8k
■ ekpcondorcentral RunningJobs  Now: 85.0   Min: 11.0   Avg: 5.4k   Max: 7.8k
```

Thanks to bwHPC and the NEMO HPC cluster team in Freiburg!

Step 5: Postprocessing

- **Checking, purging, combining:**
  - **Check again interpolation quality for individual grids**
  - **Run NNLOJET combination script → weight tables**
  - **Weighted merging of grids**
  - **Check and treat potential remaining unsuppressed fluctuations**
  - **Do some nice physics**

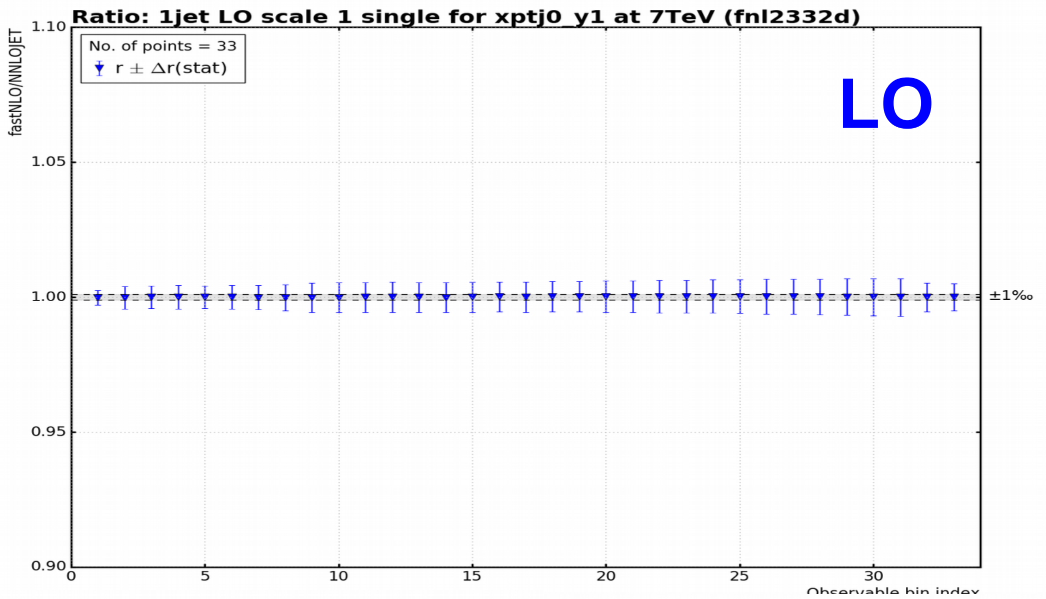
Step 6: Validation

- Check every aspect you can or you will be hit by Murphy's law!
  - Check each contribution (LO, R, V, RRa, RRb, 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/comparisons, e.g. to other programs
  - ?

Inclusive jet pT – single grid

Ratio APPLfast / NNLOJET

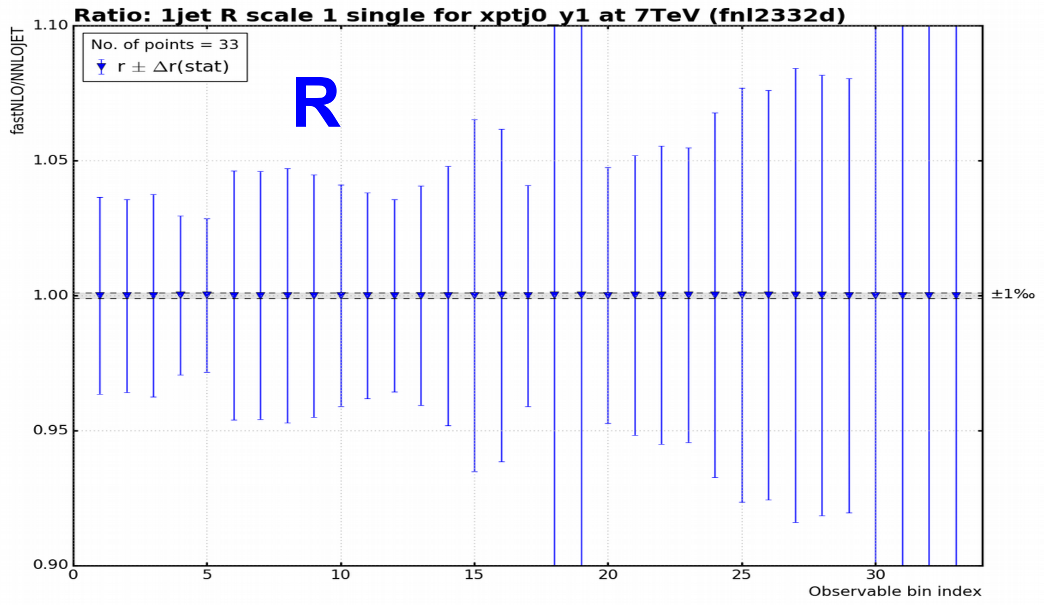
Error bars:
stat. uncertainty estimate from NNLOJET

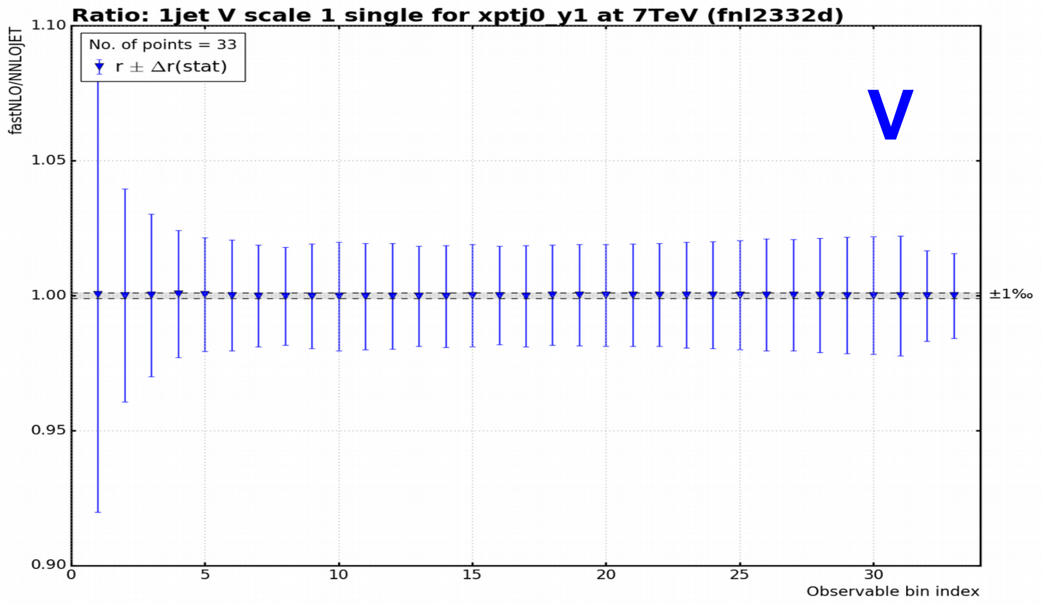


+10%

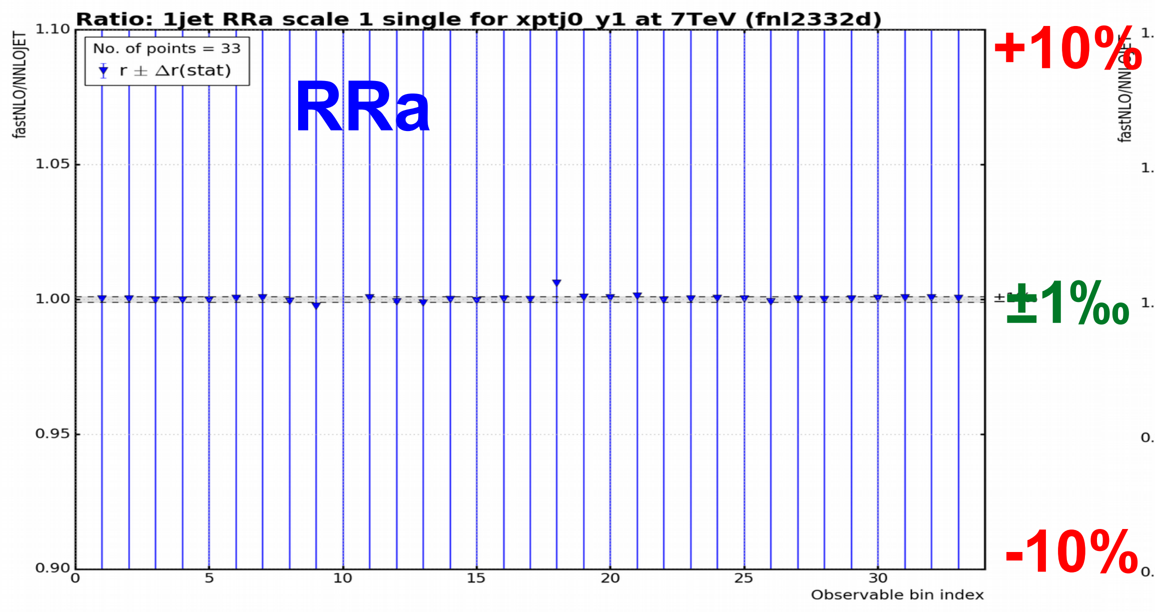
±1‰

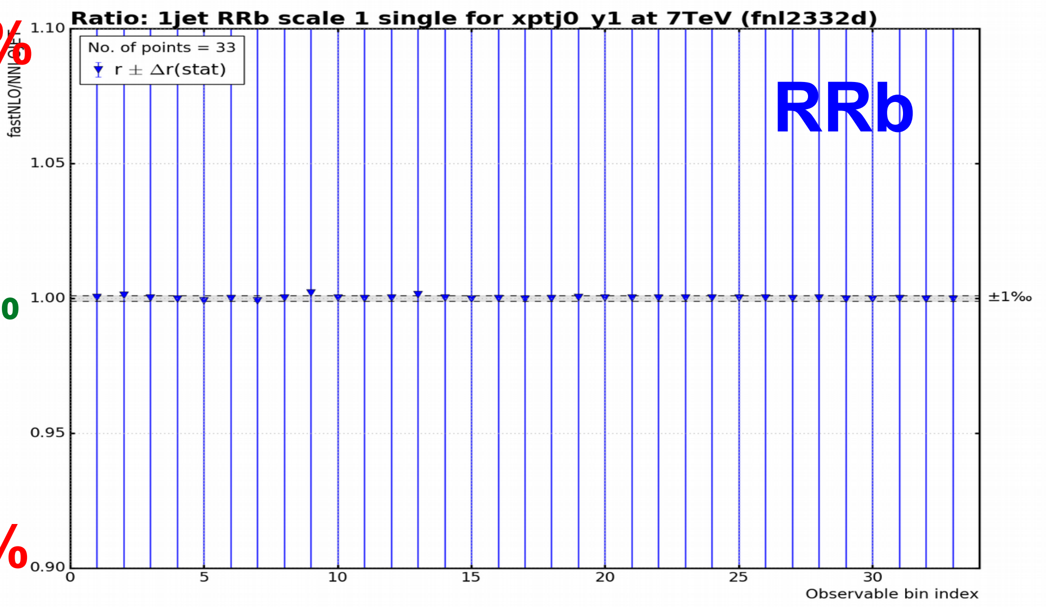
-10%

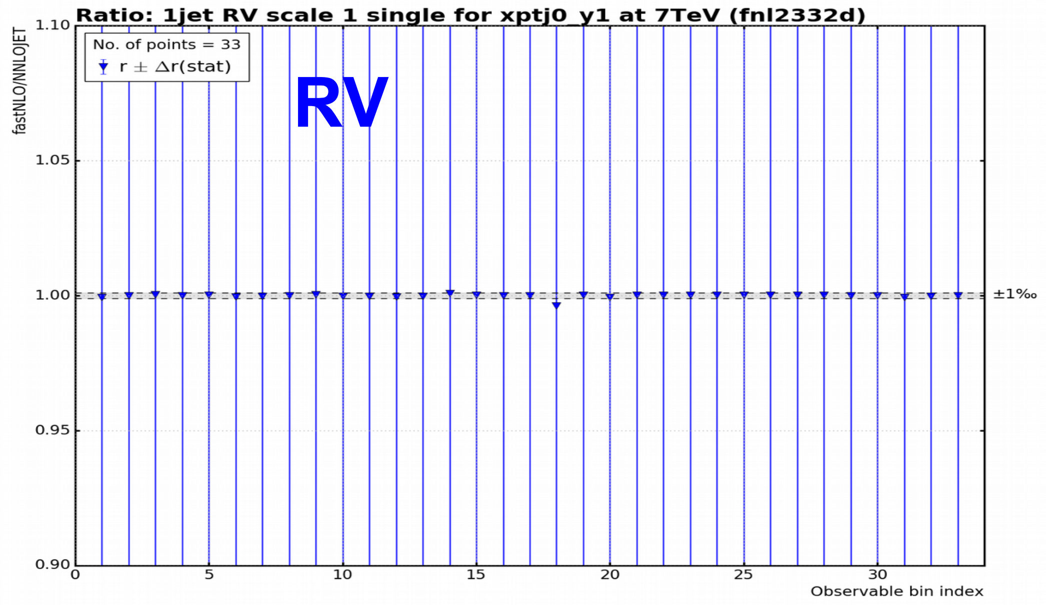


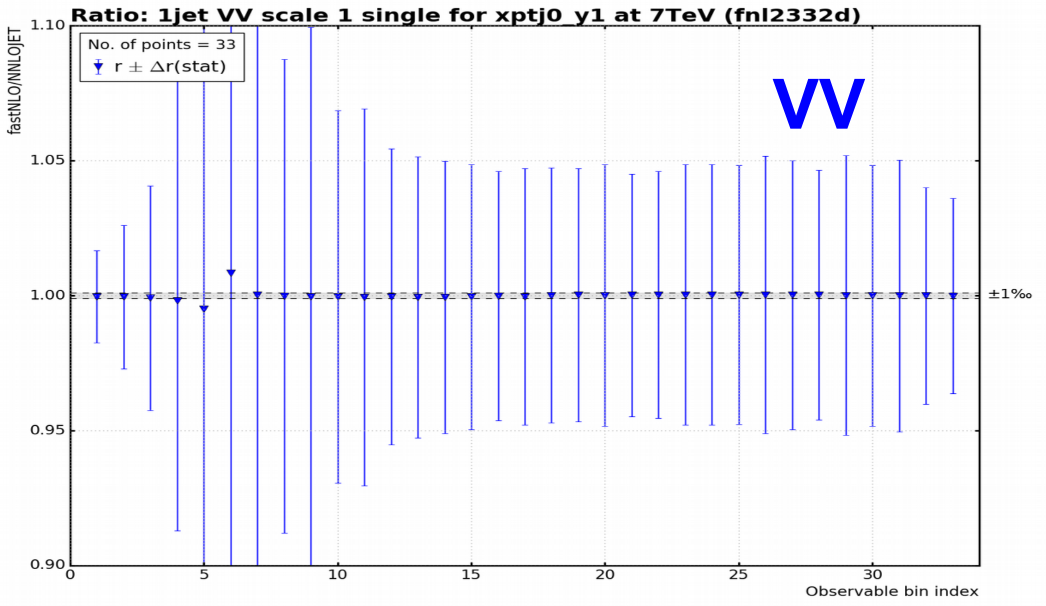


Inclusive jet pT – single grid





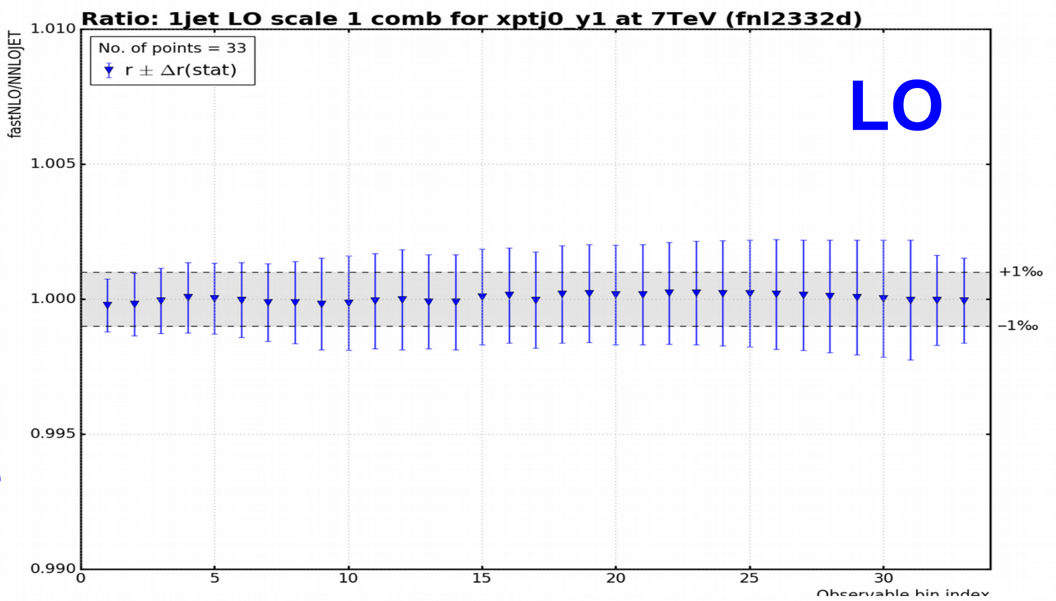




Inclusive jet pT – combined grid

Ratio FastNLO / NNLOJET

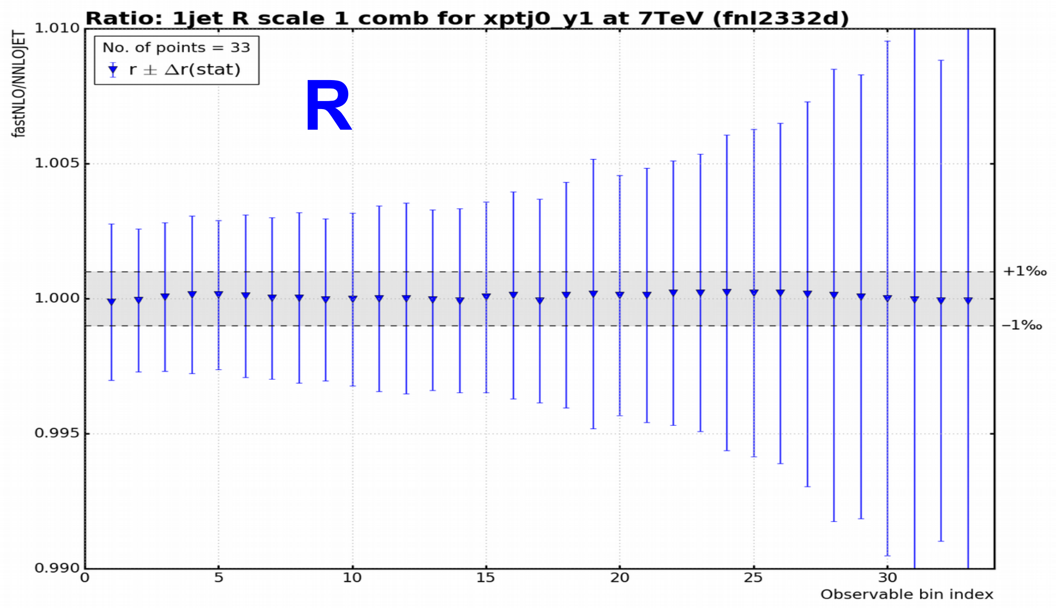
Error bars:
stat. uncertainty estimate from NNLOJET

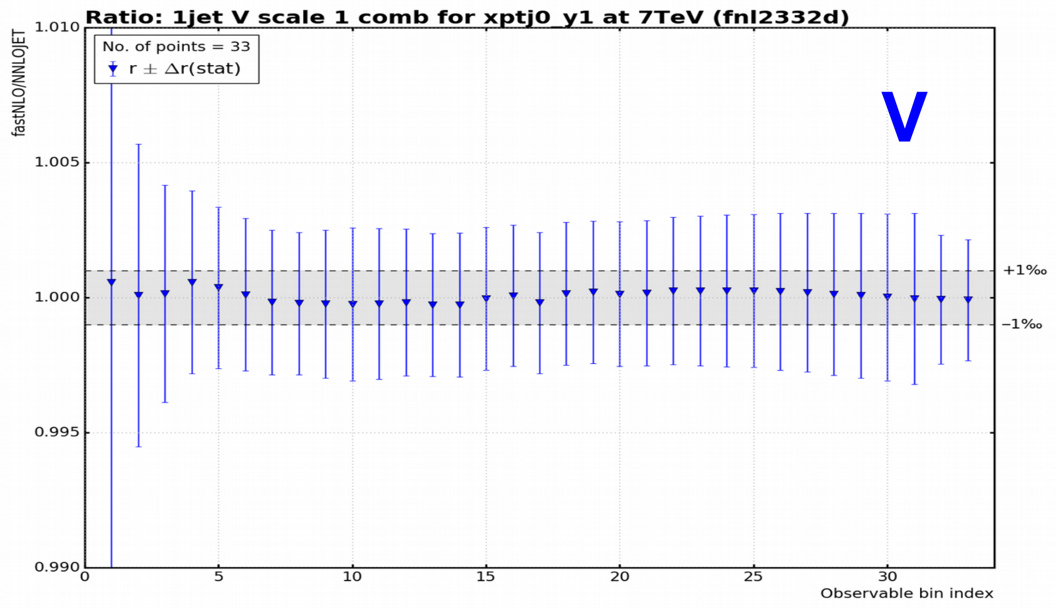


+1%

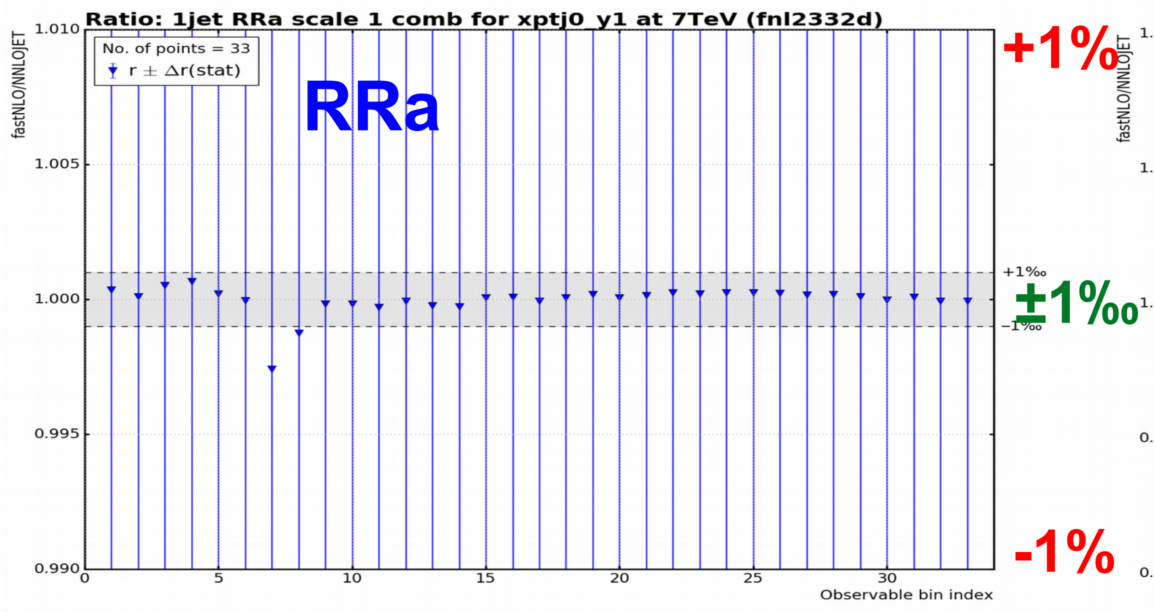
±1‰

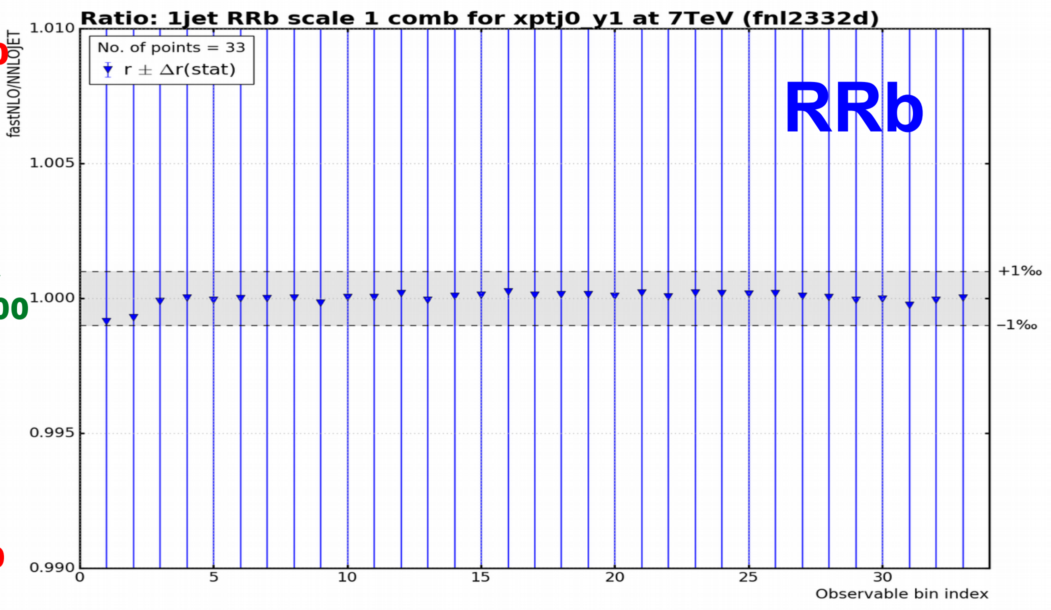
-1%

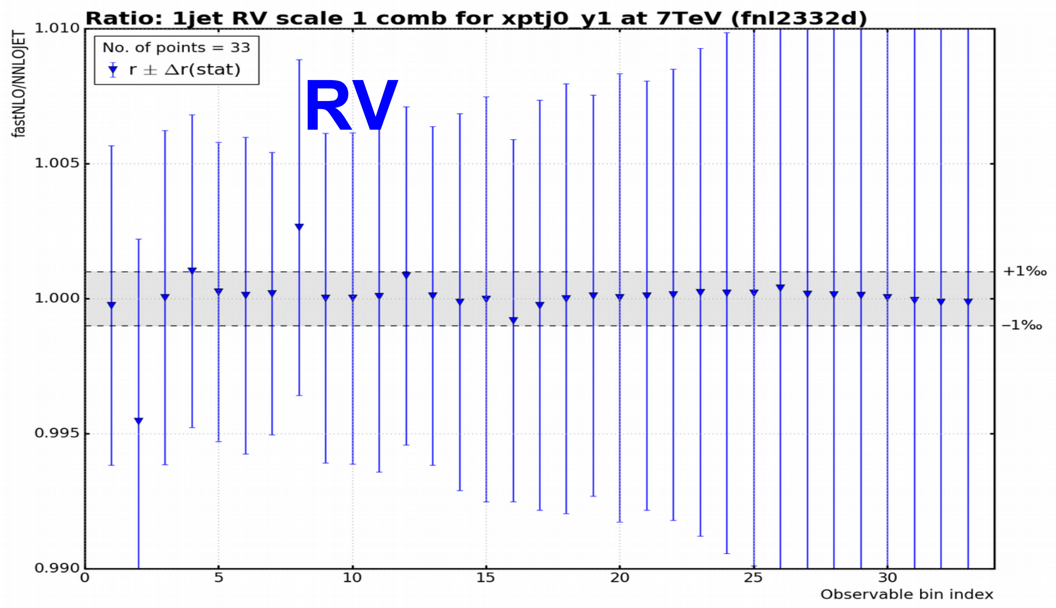


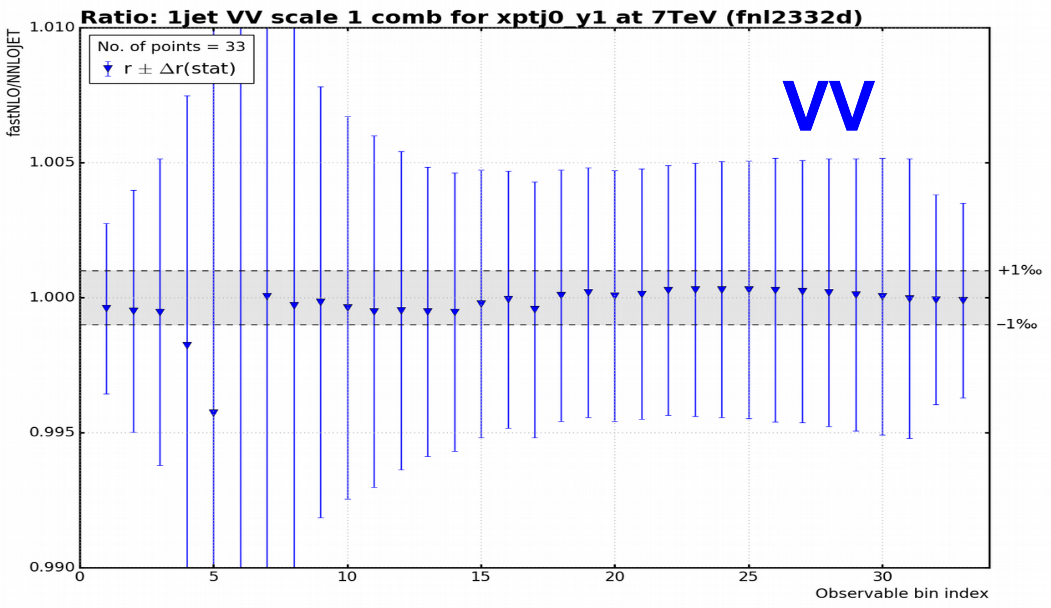


Inclusive jet pT – combined grid





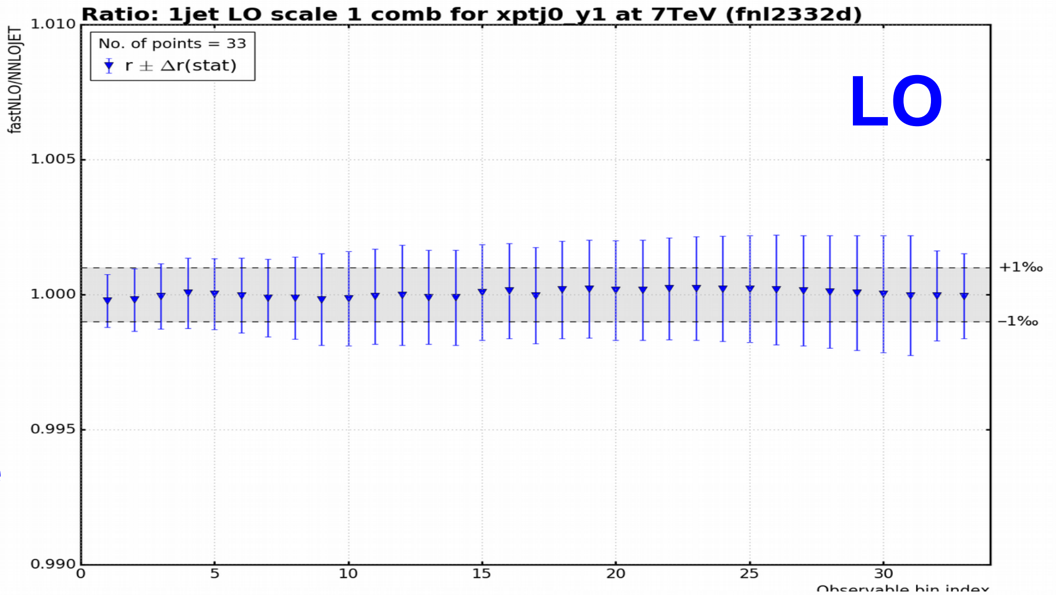




Inclusive jet pT – combined grid

Ratio FastNLO / NNLOJET

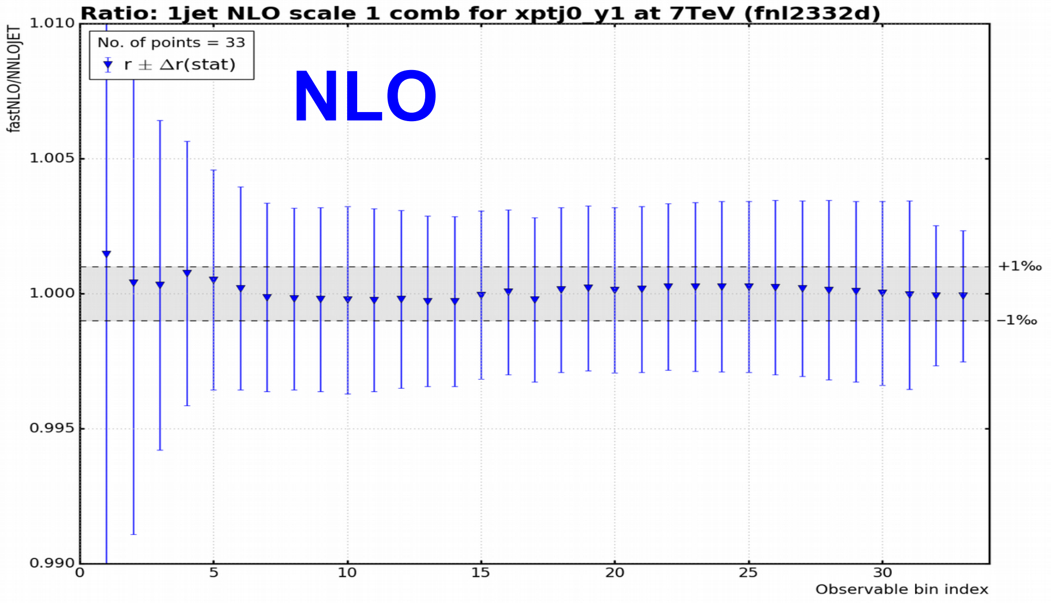
Error bars:
stat. uncertainty estimate from NNLOJET

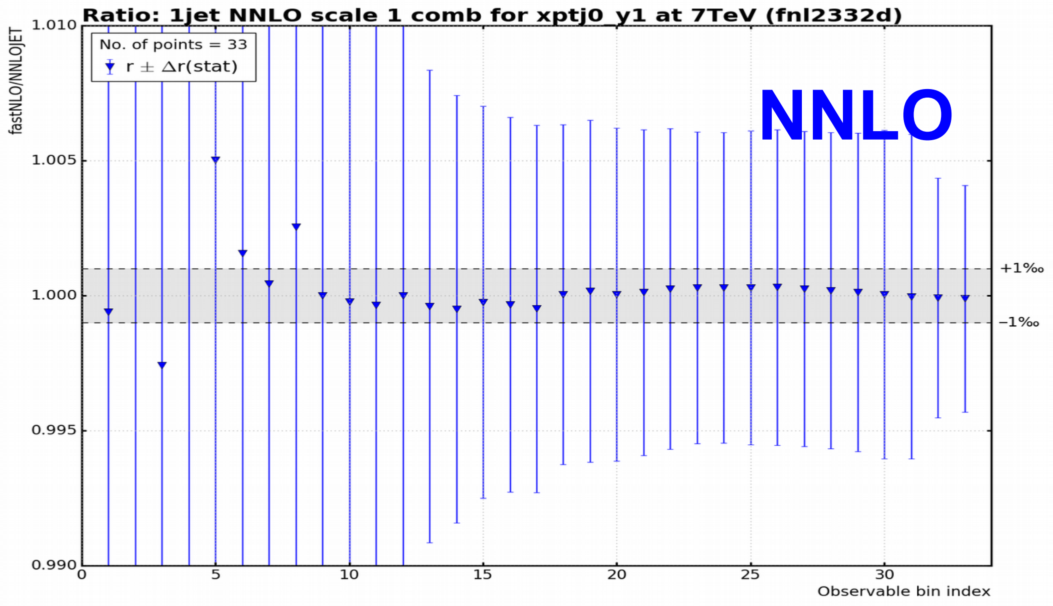


+1%

±1‰

-1%



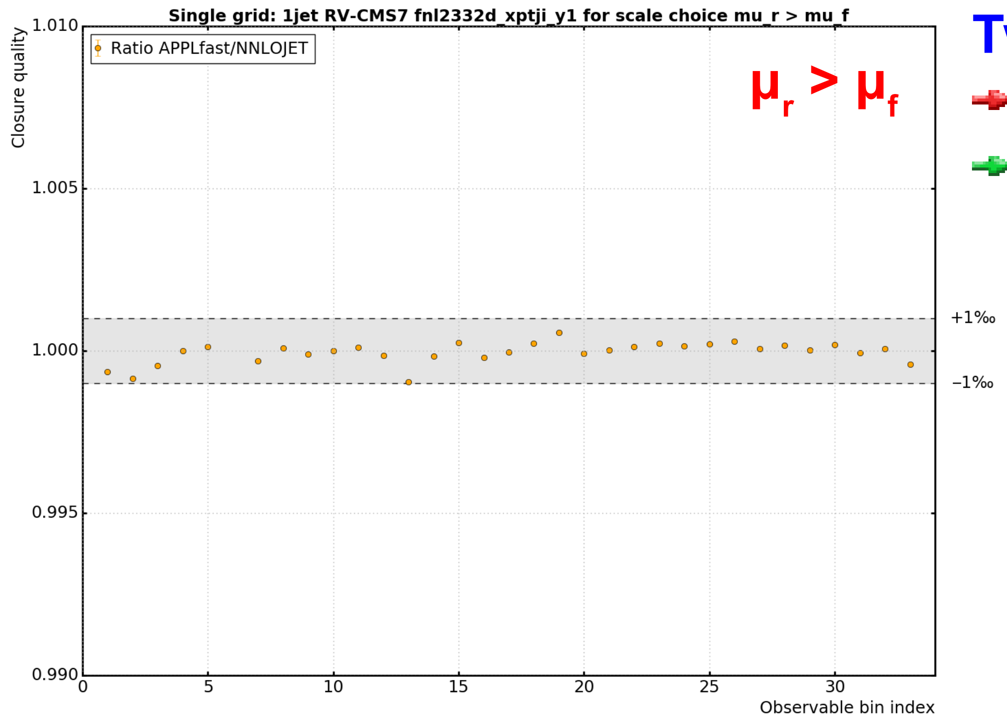


NNLO Outlook, Jan. 2018

- 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 published; code to be released, see also previous talk by A. Huss!
- APPLgrid+fastNLO interface (NNLO-Bridge) is working
- Numerous adaptations implemented by all sides
- Large-scale productions tested for Z+jet and DIS jet and pp jet
- DIS NNLO results & α_s published with H1: Eur. Phys. J. C, 2017, 77, 791
- Received final combination prescription for NNLOJET results last Nov.
  - Removes fluctuations from bad cancellations
  - fastNLO table merging implemented; looks ok → check for outliers
  - Interpolated scale variations working except asymmetric $\mu_r \neq \mu_f$ TBD
- Looking forward to many new NNLO interpolation grids in 2018

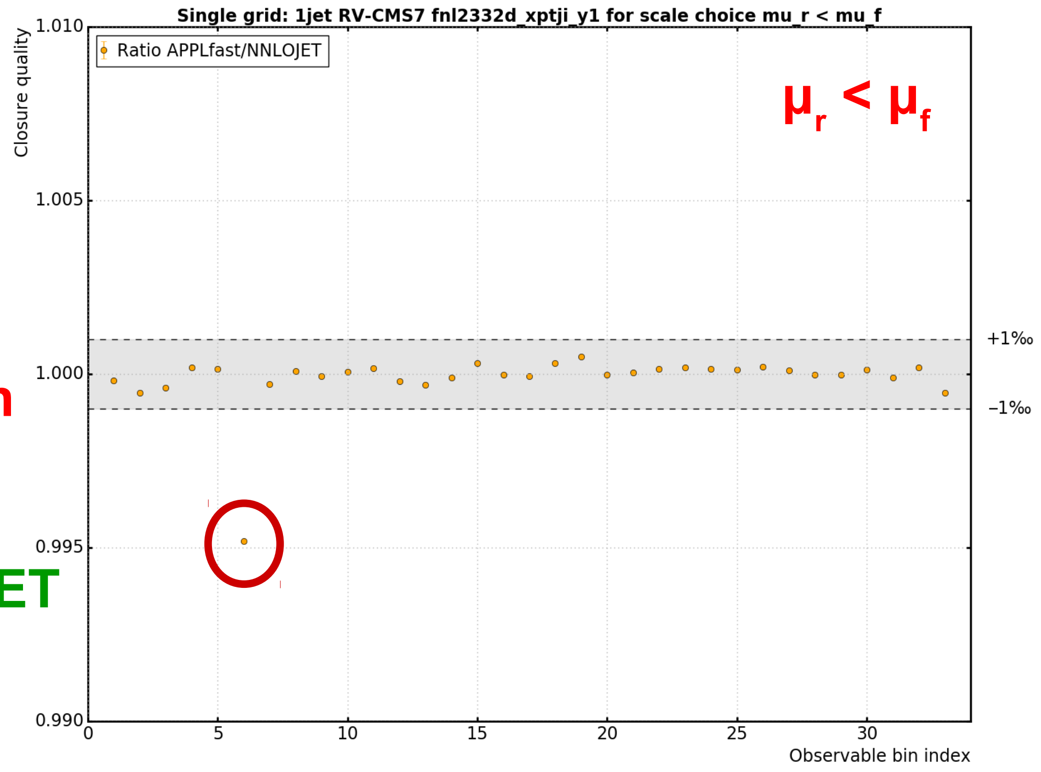
Fixed!

Single grid, scale variation, RV



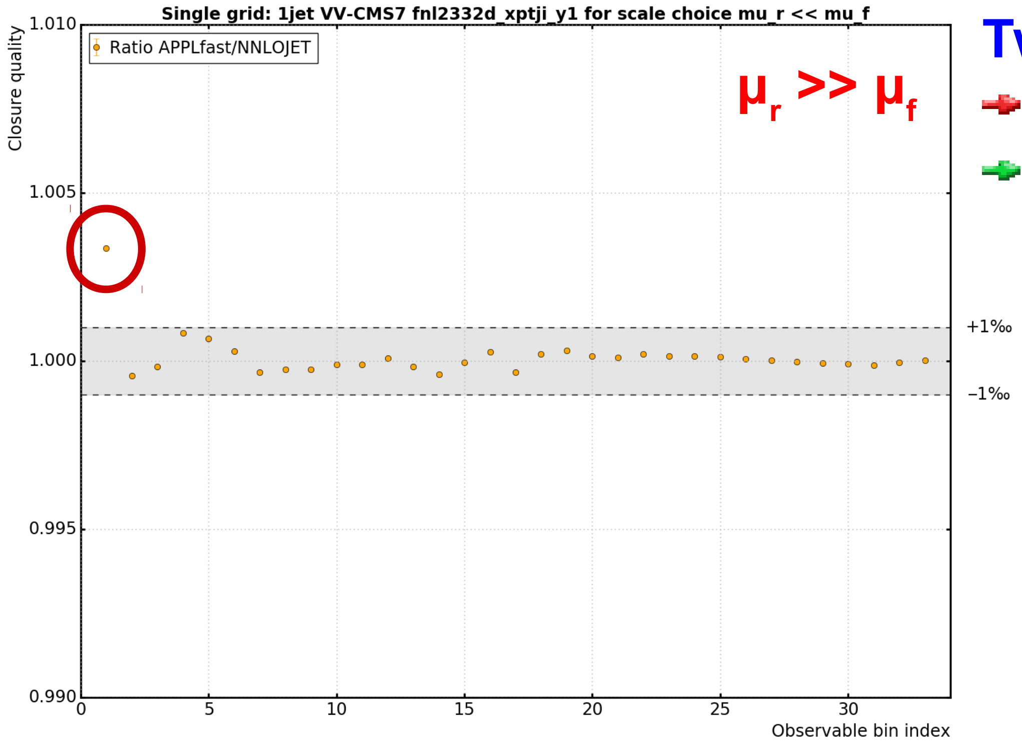
Two issues in interface to fastNLO addressed:

- **Fixed: Bug dropping weights for μ_r != μ_f**
- **No bias visible also for scale interpolation**



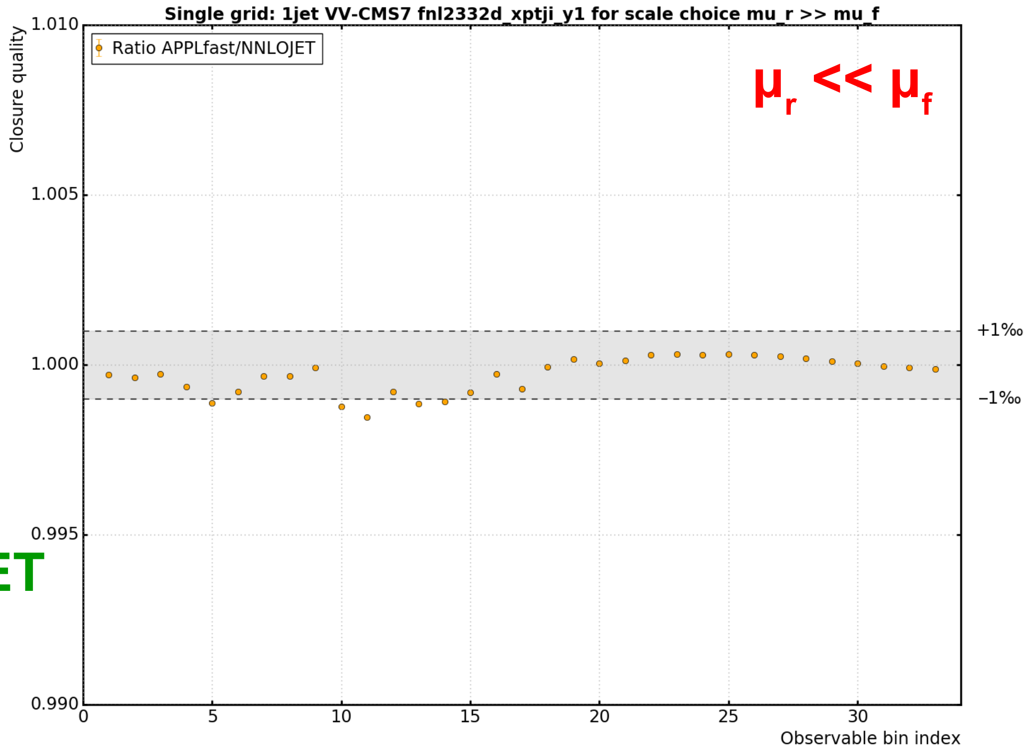
- **Numerical precision in weight summation improved**
- **Less outliers**
- **Often appear at same place as in NNLOJET $\longrightarrow$ improve with merging procedure**

Single grid, scale variation, VV



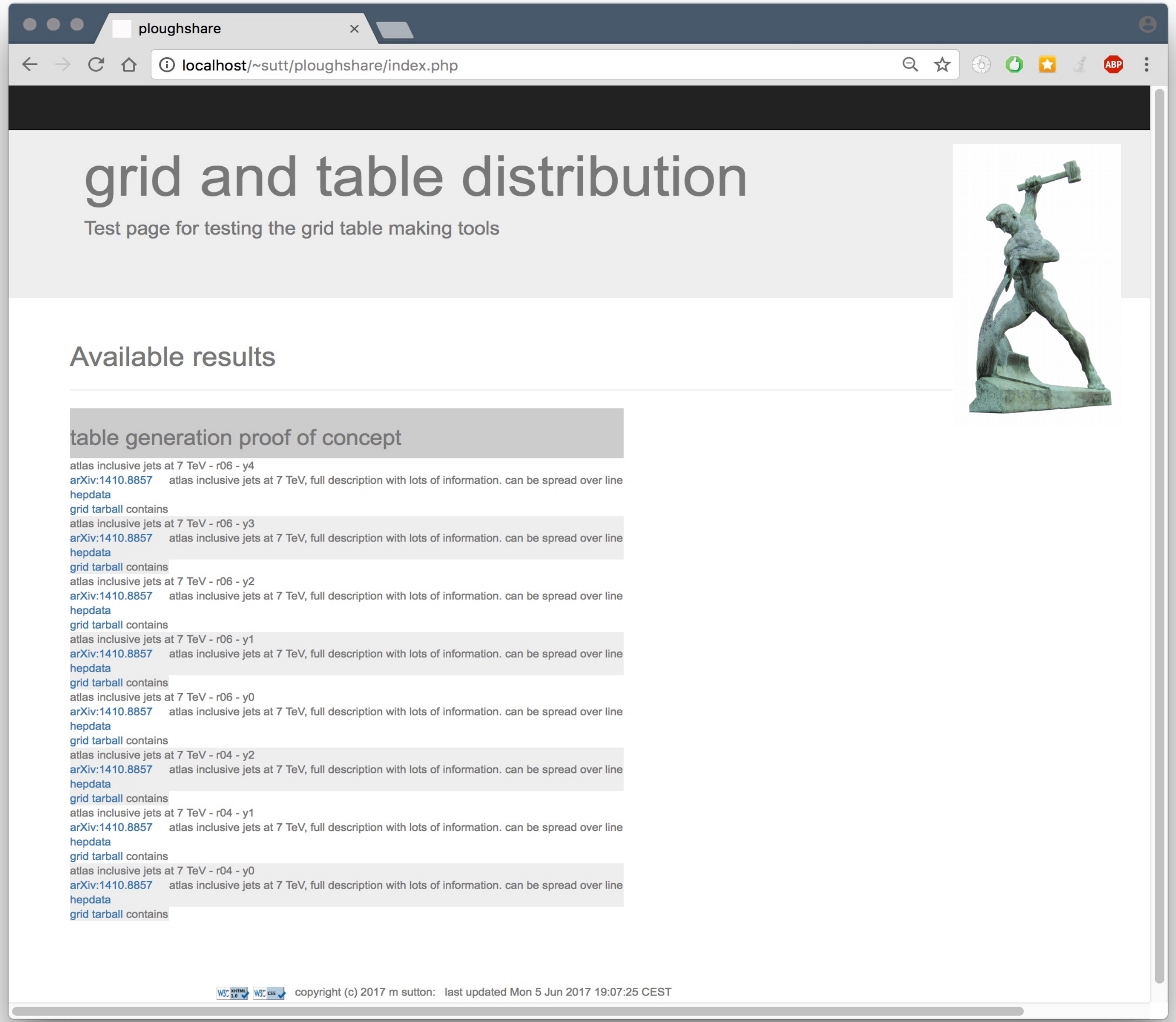
Two issues in interface to fastNLO addressed:

- **Fixed: Bug dropping weights for μ_r != μ_f**
- **No bias visible also for scale interpolation**



- **Numerical precision in weight summation improved**
- **Less outliers**
- **Often appear at same place as in NNLOJET $\longrightarrow$ improve with merging procedure**

Grid distribution – Ploughshare



grid and table distribution

Test page for testing the grid table making tools

Available results

table generation proof of concept

atlas inclusive jets at 7 TeV - r06 - y4
arXiv:1410.8857 atlas inclusive jets at 7 TeV, full description with lots of information. can be spread over line
hepdata
grid tarball contains
atlas inclusive jets at 7 TeV - r06 - y3
arXiv:1410.8857 atlas inclusive jets at 7 TeV, full description with lots of information. can be spread over line
hepdata
grid tarball contains
atlas inclusive jets at 7 TeV - r06 - y2
arXiv:1410.8857 atlas inclusive jets at 7 TeV, full description with lots of information. can be spread over line
hepdata
grid tarball contains
atlas inclusive jets at 7 TeV - r06 - y1
arXiv:1410.8857 atlas inclusive jets at 7 TeV, full description with lots of information. can be spread over line
hepdata
grid tarball contains
atlas inclusive jets at 7 TeV - r06 - y0
arXiv:1410.8857 atlas inclusive jets at 7 TeV, full description with lots of information. can be spread over line
hepdata
grid tarball contains
atlas inclusive jets at 7 TeV - r04 - y2
arXiv:1410.8857 atlas inclusive jets at 7 TeV, full description with lots of information. can be spread over line
hepdata
grid tarball contains
atlas inclusive jets at 7 TeV - r04 - y1
arXiv:1410.8857 atlas inclusive jets at 7 TeV, full description with lots of information. can be spread over line
hepdata
grid tarball contains
atlas inclusive jets at 7 TeV - r04 - y0
arXiv:1410.8857 atlas inclusive jets at 7 TeV, full description with lots of information. can be spread over line
hepdata
grid tarball contains

copyright (c) 2017 m sutton: last updated Mon 5 Jun 2017 19:07:25 CEST

- **Where to find the grids?**
  Idea: New hep forge package where registered users upload grids with some documentation
- **Registered(!) user gets FAME or BLAME**
- **Automated job treats the upload:**
  - **Add to the appropriate location in the file system**
  - **Generate relevant lists, and display web pages**
- **Should provide a user interface for automated download with a simple line of code**
- **Have expression of interest from many stakeholders …**
  - **ATLAS, CMS, HERA (H1 + ZEUS), MMHT, CTEQ, NNPDF, xFitter, APPLgrid, fastNLO …**
- **Simple proof of concept is there, development stalled a bit lately by lack of time. HELP is WELCOME!**

Summary

- APPLfast interface (NNLO-Bridge) and interpolation is working
- Large-scale productions tested for Z+jet, DIS jet, and pp jets
- Combination of grids with weights à la NNLOJET implemented
- Address last issues and check on possible remaining outliers in grids even with weighted combination
- No essential changes in xFitter needed, but some updates will become necessary (One fastNLO update already in → Engin's talk)
- Start 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 on HepForge, open for contributions from the community

Thank you for your attention!

Backup

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

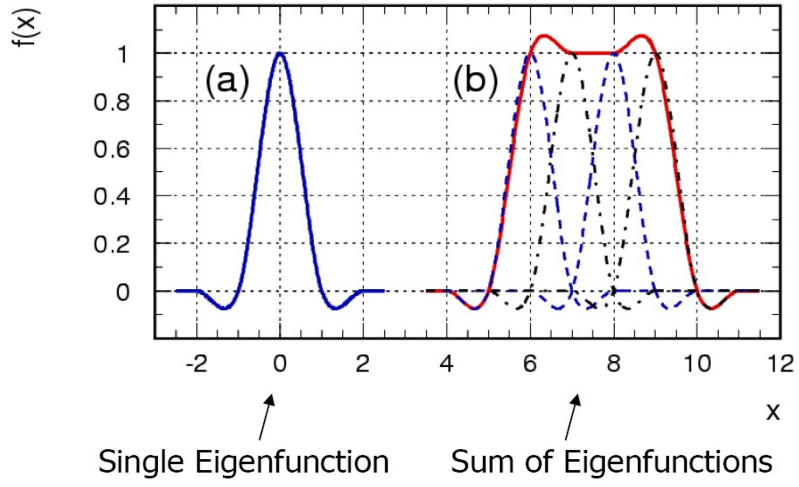
- Introduce set of n discrete x-nodes, x_i's being equidistant in a function f(x)
- Take set of Eigenfunctions $E_i(x)$ around nodes x_i

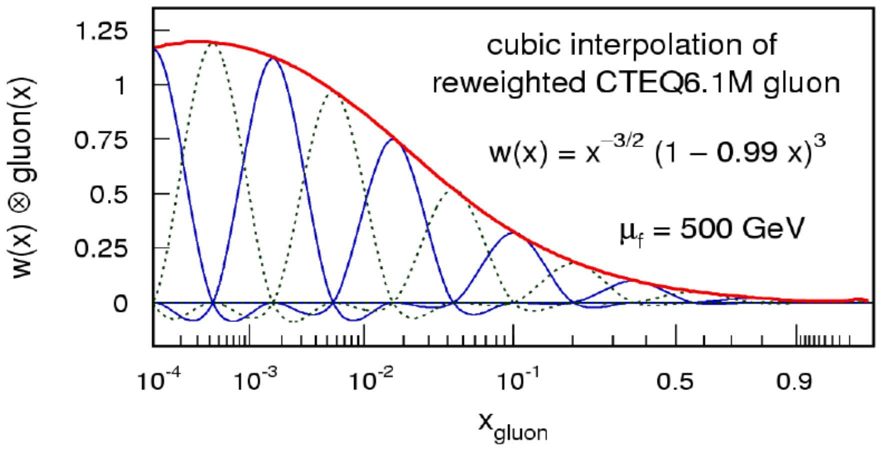
→ Interpolation kernels
- Actually a rather old idea, see e.g.
  C. Pascaud, F. Zomer (Orsay, LAL), LAL-94-42

→ 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

Flexible-scale tables

- Storage of scale-independent weights enable full scale flexibility also in NNLO
  - Additional logs in NNLO

$$\omega(\mu_R,\mu_F) = \underbrace{\omega_0 + \log(\mu_R^2)\omega_R + \log(\mu_F^2)\omega_F}_{\text{log's for NLO}} + \underbrace{\log^2(\mu_R^2)\omega_{RR} + \log^2(\mu_F^2)\omega_{FF} + \log(\mu_R^2)\log(\mu_F^2)\omega_{RF}}_{\text{additional log's in NNLO}}$$

  - Store weights: w_0, w_R, w_F, w_{RR}, w_{FF}, w_{RF} for order α_s^{n+2} contributions

- Advantages
  - Renormalization and factorization scale can be varied *independently* and by *any* factor
    - No time-consuming 're-calculation' of splitting functions in NLO necessary
  - Only small increase in amount of stored coefficients

- Implementation
  - *Two* different observables can be used for the scales
    - e.g.: H_T and $p_{T,max}$
    - or e.g.: p_T and $|y|$
    - …
  - *Any function* of those *two observables* can be used for calculating scales