

Application of fastNLO to NNLO calculations

*fast*NLO

Daniel Britzger, Marco Guzzi, Klaus Rabbertz, Georg Sieber, Fred Stoiber, Markus Wobisch
(DESY, DESY, KIT, KIT, KIT, Louisiana Tech)

DIS 2014

XXII. International Workshop on Deep-Inelastic Scattering and Related Subjects

April 30, 2014



Motivation

Interpretation of experimental data relies on

- Availability of reasonably fast theory calculations
- Often needed: Repeated computation of (almost) same cross sections
 - Observables, binning, phase space given by experimental data

Examples for a specific analysis:

- Use of various PDFs (CT, MSTW, NNPDF, ...) for data/theory comparison
- Determine PDF uncertainties
- Derivation of scale uncertainties
- Use data set in fit of PDFs and/or $\alpha_s(M_Z)$

Sometimes NLO predictions can be computed fast

But some are very slow

e.g. jet cross sections, VB+jets, Drell-Yan, ...

Need procedure for fast repeated computations of NLO cross sections

- Use fastNLO (in use by most PDF fitting groups)

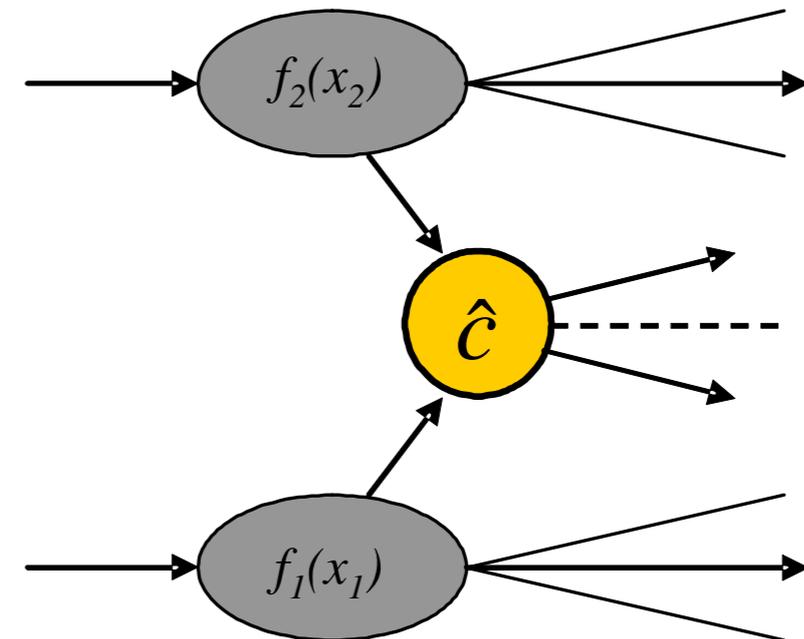
Basic working principle

Cross section in hadron-hadron collisions in pQCD

- Many cross section calculations are time consuming (e.g. jets)

$$\sigma = \sum_{a,b,n} \int_0^1 dx_1 \int_0^1 dx_2 \alpha_s^n(\mu_r) \cdot c_{a,b,n}(x_1, x_2, \mu_r, \mu_f) \cdot f_{1,a}(x_1, \mu_f) f_{2,b}(x_2, \mu_f)$$

- strong coupling α_s in order n
- PDFs of two hadrons f_1, f_2
- Parton flavors a, b
- perturbative coefficient $c_{a,b,n}$
- renormalization and factorization scales μ_r, μ_f
- momentum fractions x_1, x_2



PDF and α_s are external input

Perturbative coefficients are independent from PDF and α_s

Idea: factorize PDF, α_s and scale dependence

The fastNLO concept

Introduce interpolation kernel

- 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

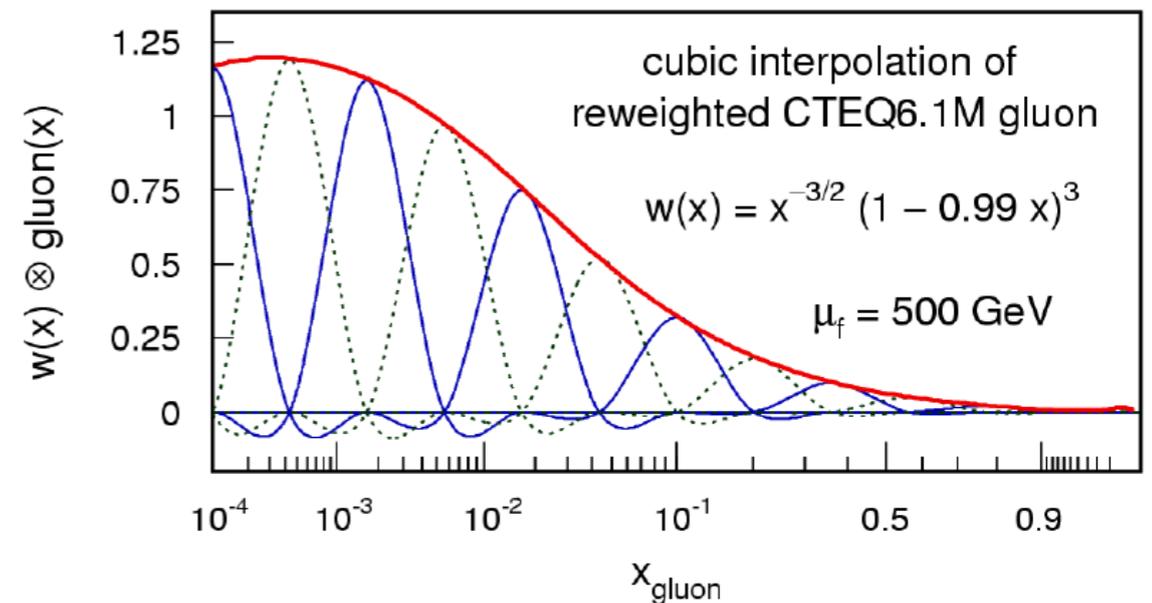
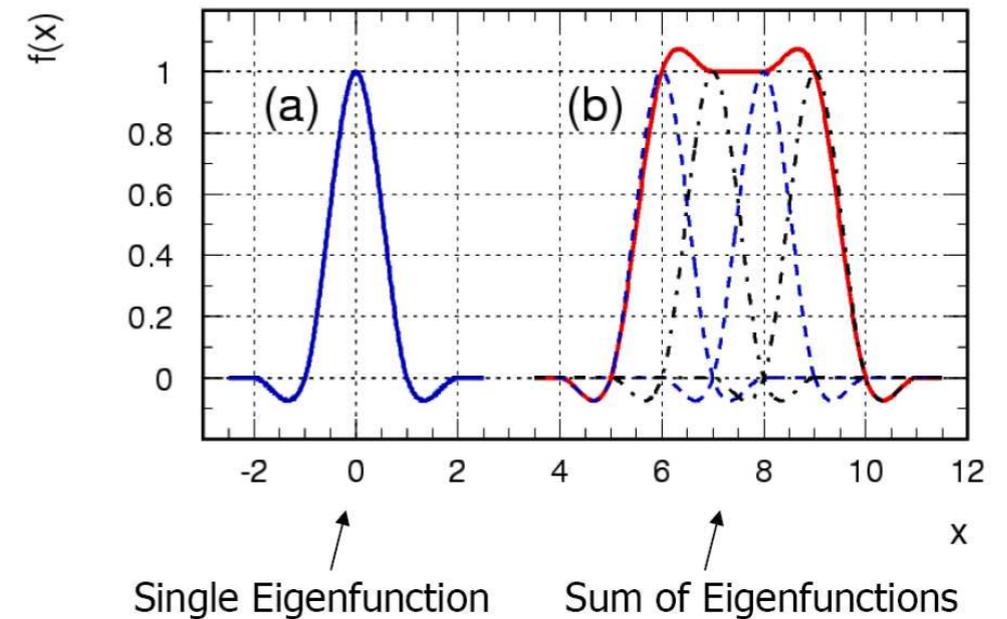
Single PDF is replaced by a linear combination of interpolation kernels

Improve interpolation by reweighting PDF

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

Scale dependence

- Similar interpolation procedure also for scales
- Interpolation nodes in x and scales are stored together in look-up table
- Two different observables can be chosen as scale in one table



Convolution integrals become discrete sums
=> Values of perturbative coefficients can be stored in a table

Calculations with fastNLO in NNLO

Problem

- Scale variations become more difficult in NNLO than in NLO

Current available implementations for NLO calculations

Renormalization scale variations

- Scale variations applying RGE
 - Use LO matrix elements times $n\beta_0\ln(c_r)$ [*fastNLO, APPLgrid* (EPJ C (2010) 66: 503)]
- Flexible-scale implementation
 - Store scale-independent weights: $\omega(\mu_R, \mu_F) = \omega_0 + \log(\mu_R)\omega_R + \log(\mu_F)\omega_F$ [*fastNLO*]

Factorization scale variations

- Calculate LO DGLAP splitting functions using HOPPET [*APPLgrid*]
- Store coefficients for desired scale factors [*fastNLO*]
- Flexible-scale implementation [*fastNLO*]

Scale variations for NNLO calculations

- a-posteriori renormalization scale variations become more complicated
- NLO splitting functions are needed for factorization scale variations
 - Calculations become slow again => Not desired for fast repeated calculations

Flexible-scale implementation in NNLO

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

fastNLO 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

'Flexible-scale concept': Best choice for performant NNLO calculations

Application to differential ttbar cross sections in approx. NNLO

Application of flexible-scale concept to NNLO calculations

Interface to DiffTop code

- DiffTop
 - Code for calculation of heavy quark production within threshold resummation formalism in Mellin space
 - See talk by M.Guzzi
- Differential ttbar cross sections in approx. NNLO
 - $d\sigma/dp_T$
 - $d\sigma/dy$

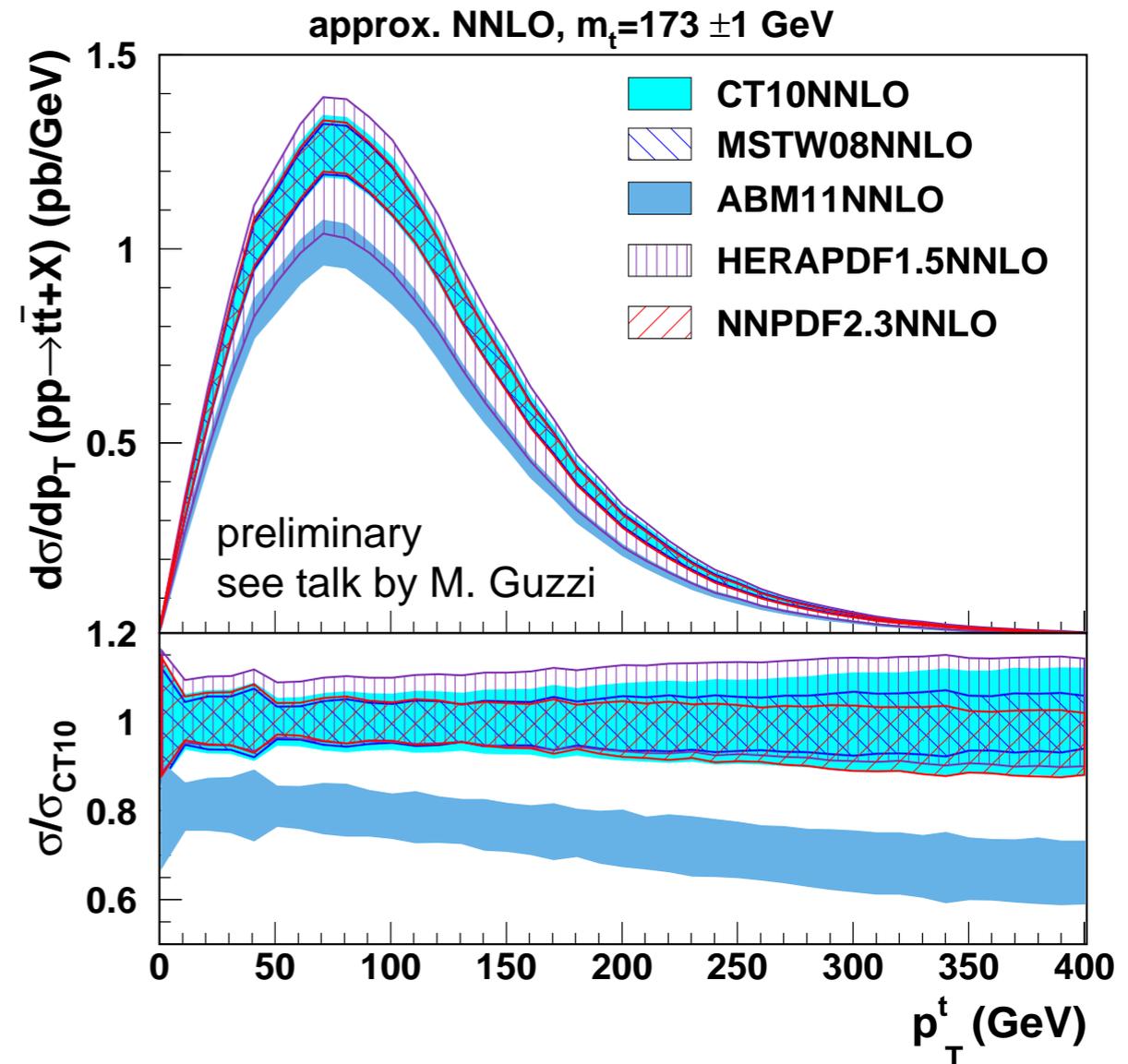
Benefit in speed

- NNLO calculation O(days-weeks)
- fastNLO calculation O(<1s)

fastNLO facilitates study of PDF dependence

Particularly including PDF uncertainties

- 262 re-calculations are required



786 repeated calculations needed including variation of m_t

Application to differential $t\bar{t}$ cross sections in approx. NNLO

Without recalculating the coefficients

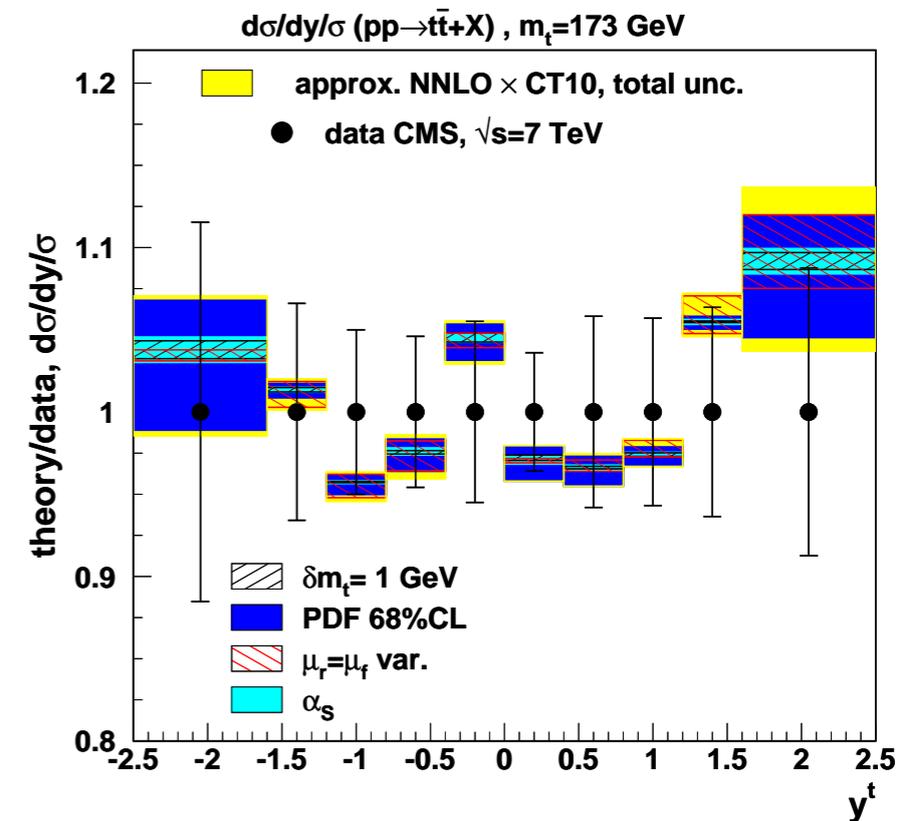
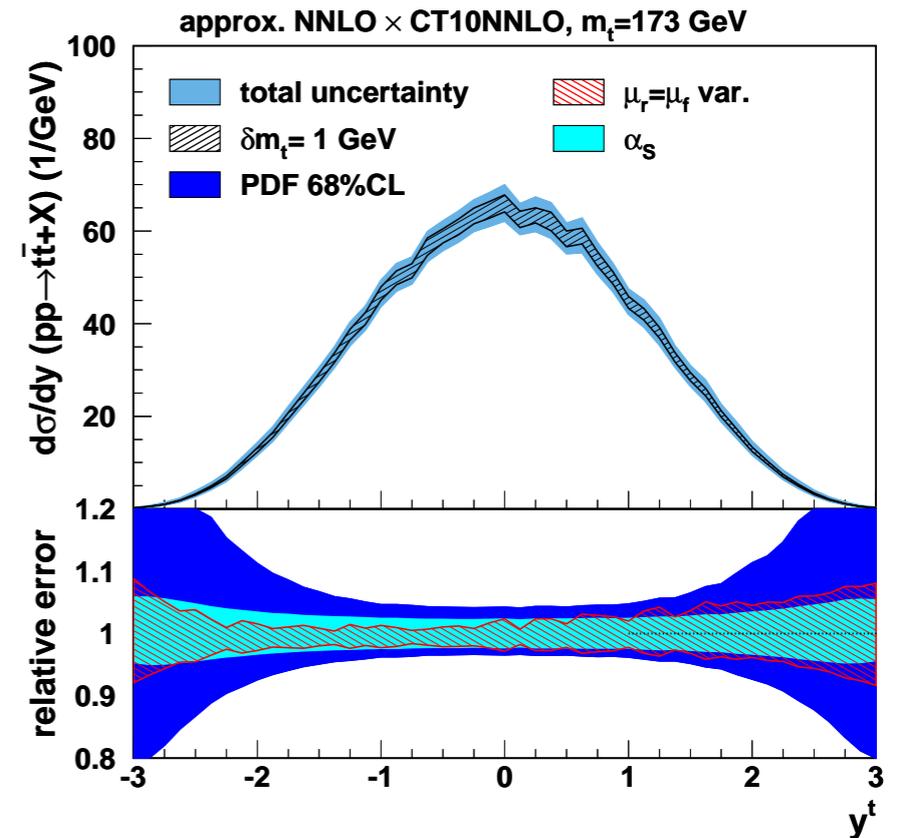
- Variation of the scales within milliseconds
- Variation of α_s
- Determination of PDF uncertainties
- Also choice of the scales possible
here: $\mu_{r/f} = f(p_T, y, m_t)$

Variation of m_t

- m_t is a third hard scale in this process
- m_t is not factorized in the current approach
 - Separate fastNLO tables have been computed for different values of m_t

Fast recomputation of cross sections for a given measurement enables application of time-consuming (N)NLO calculations to PDF and/or α_s -fits

- Large gluon uncertainties at high-x can be reduced using $t\bar{t}$ cross sections



Accuracy of fastNLO interpolation in NNLO

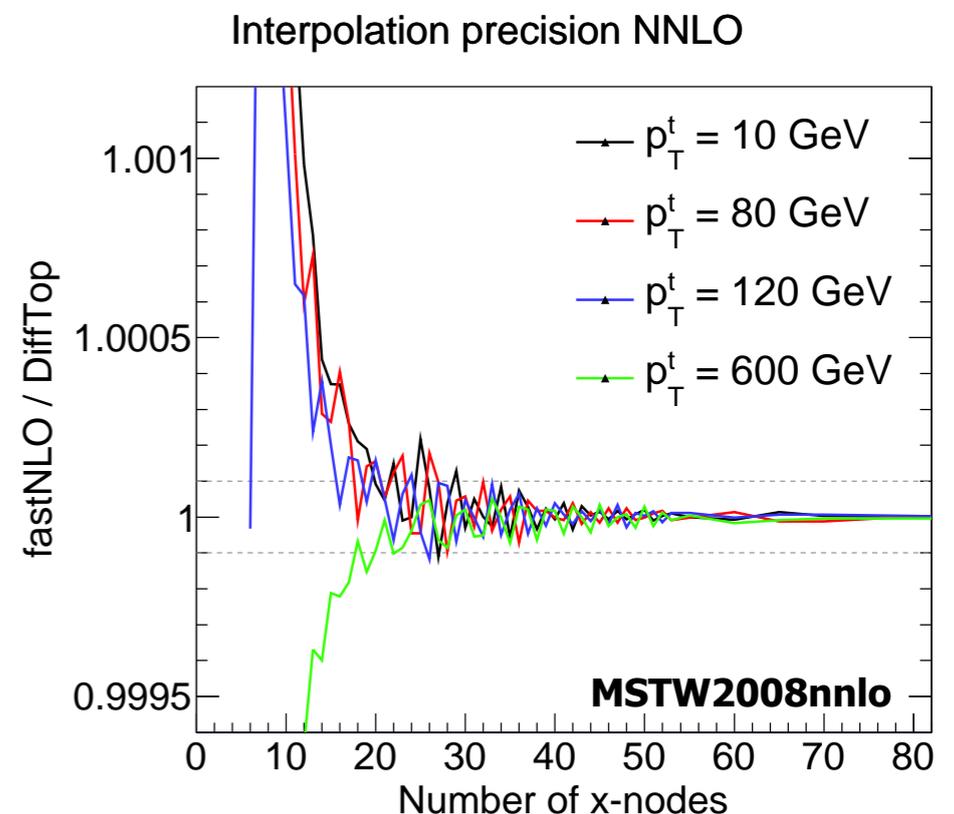
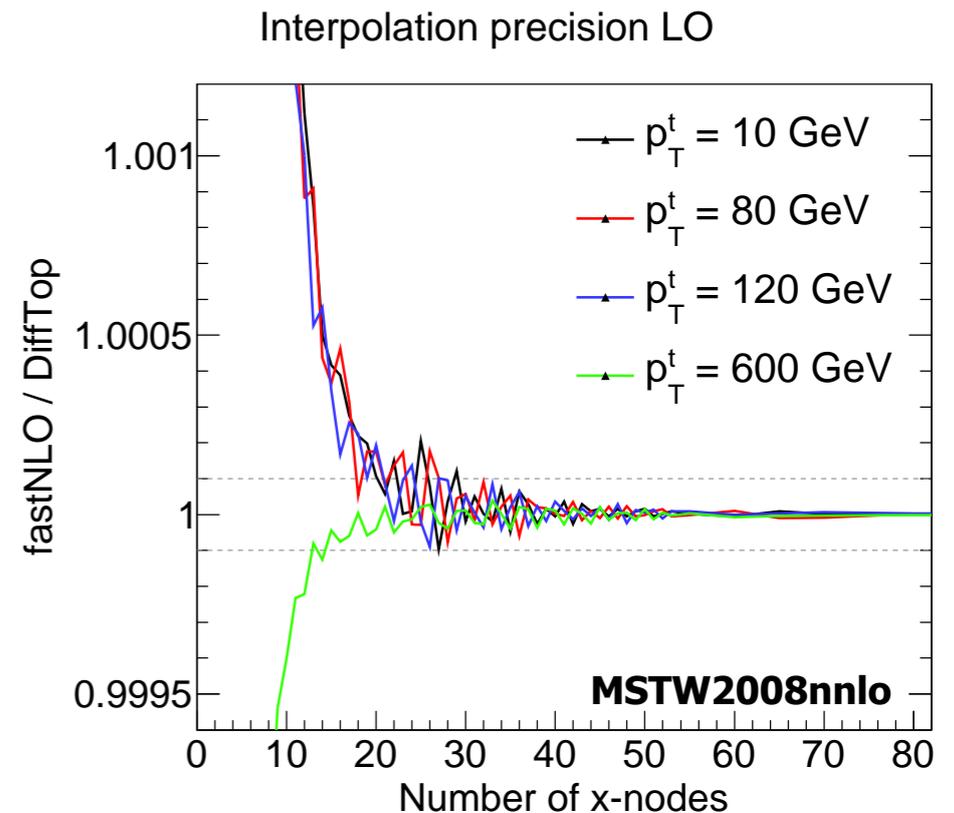
Compare cross sections calculated with DiffTop standalone compared to fastNLO

- Interpolation accuracy depends on number of nodes and on chosen interpolation kernel
- Bicubic interpolation kernels used
- Compare contribution order by order separately
- Data probe x-range of $2 \cdot 10^{-3} < x < 1$
 - High x-range has more distinct PDF shapes

fastNLO/DiffTop

- Perfect agreement within numerical precision reachable ($O(10^{-6})$)
- NNLO has same accuracy as LO
- With 18 nodes agreement better than $2 \cdot 10^{-4}$
- Accidental ‘interference effects’ with PDF grid may cause small fluctuations $O(2 \cdot 10^{-4})$
 - > Numerical uncertainty of PDF grids

fastNLO interpolation does not introduce numerical biases



New tool: fastNLO toolkit

What about application of fastNLO to other processes/programs ?

Hardly any theoretical limitation of fastNLO concept to pQCD or EW calculations

Why not used more frequently?

Interface of fastNLO to theory programs often very complicated...

- Theory codes are not optimized (at all) for fastNLO
- Technical difficulties are mostly limiting factor in usage

Goal:

Provide simple and flexible code to interface fastNLO to any kind of (N)NLO program

Newly developed tool: *fast*NLO Toolkit



Application procedure of new 'fastNLO Toolkit'

fastNLO needs access to

- Matrix elements before convolution with PDFs
- x-values where PDFs are evaluated
- Observables
- Scale definitions

Various NLO and NNLO programs have very different software architecture

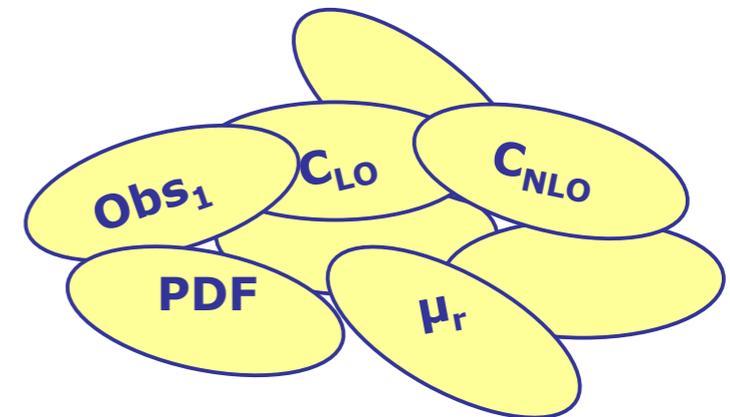
Application procedure of new 'fastNLO Toolkit'

fastNLO needs access to

- Matrix elements before convolution with PDFs
- x-values where PDFs are evaluated
- Observables
- Scale definitions

Various NLO and NNLO programs have very different software architecture

NLO program A



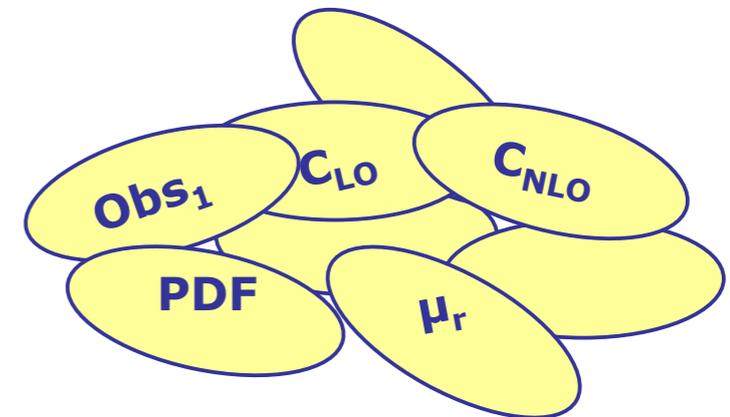
Application procedure of new 'fastNLO Toolkit'

fastNLO needs access to

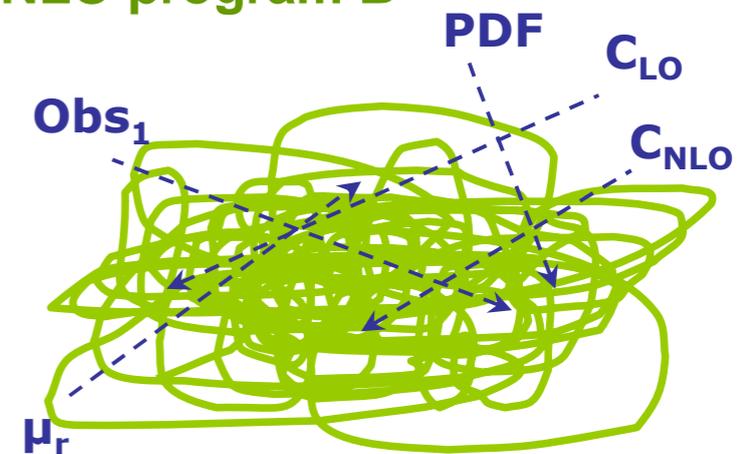
- Matrix elements before convolution with PDFs
- x-values where PDFs are evaluated
- Observables
- Scale definitions

Various NLO and NNLO programs have very different software architecture

NLO program A



NLO program B



Application procedure of new 'fastNLO Toolkit'

fastNLO needs access to

- Matrix elements before convolution with PDFs
- x-values where PDFs are evaluated
- Observables
- Scale definitions

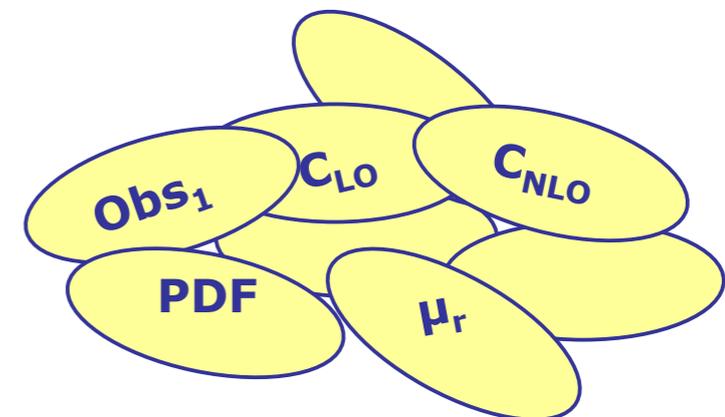
Various NLO and NNLO programs have very different software architecture

Reasons

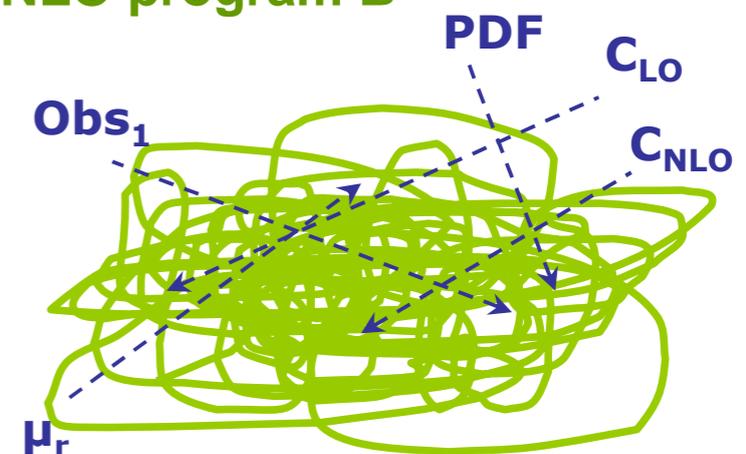
- Optimized for efficient calculation
- Straight implementation of math. formulae
- Historically grown codes
- Usage of well established algorithms (e.g. vegas)

(N)NLO programs often look to non-authors like different kind of pasta

NLO program A

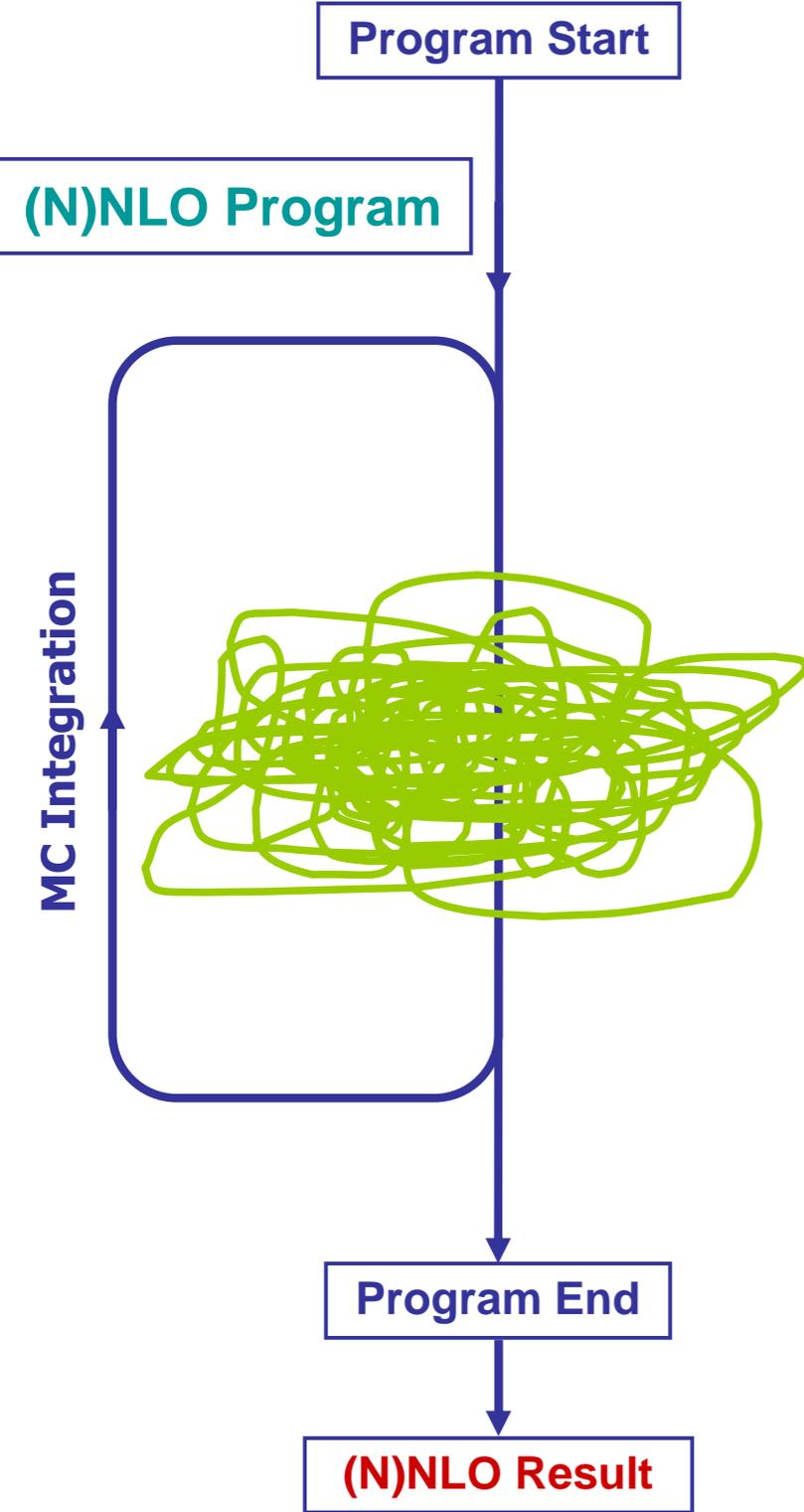


NLO program B

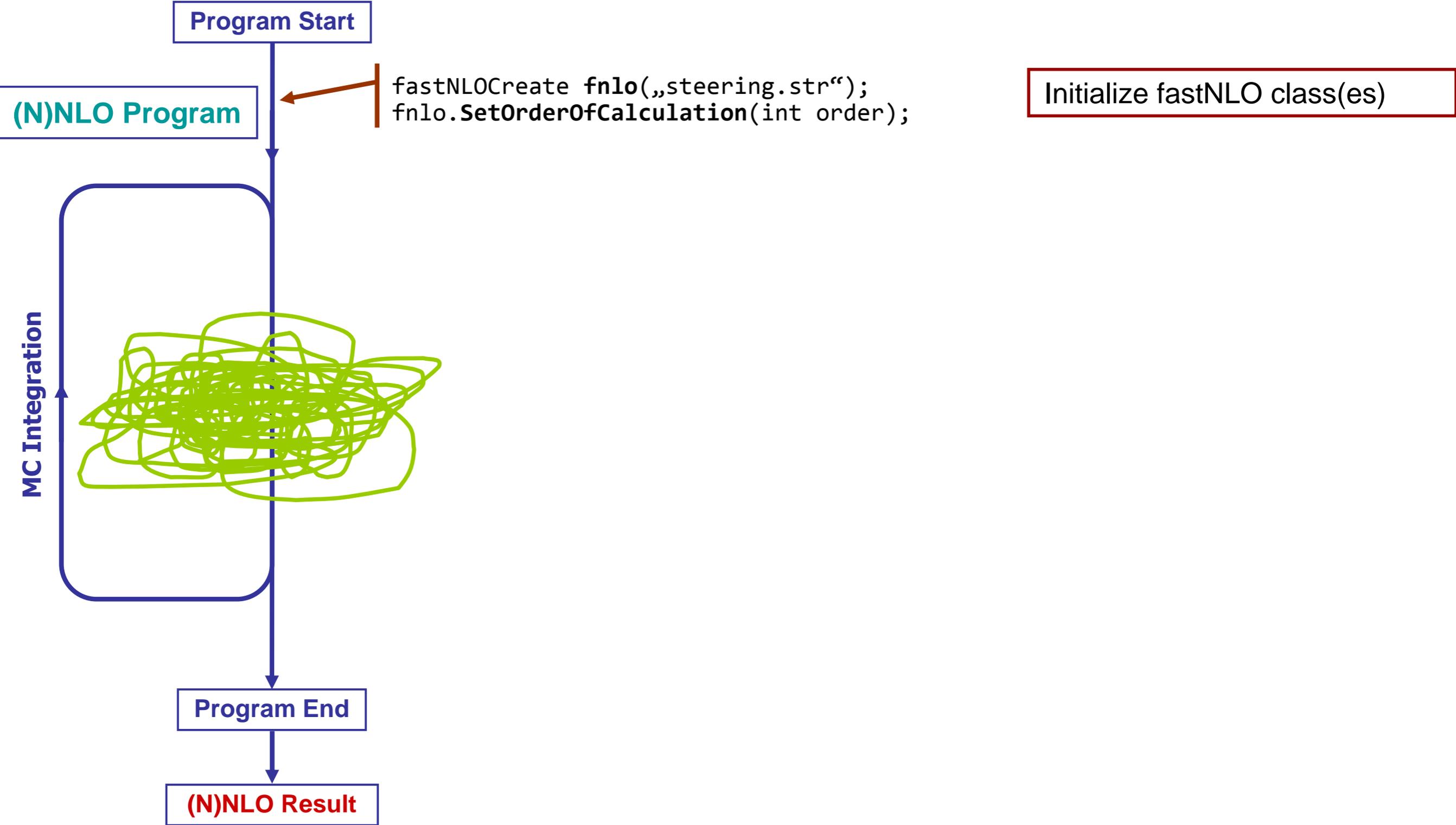


Think about a general interface to any kind of theoretical program

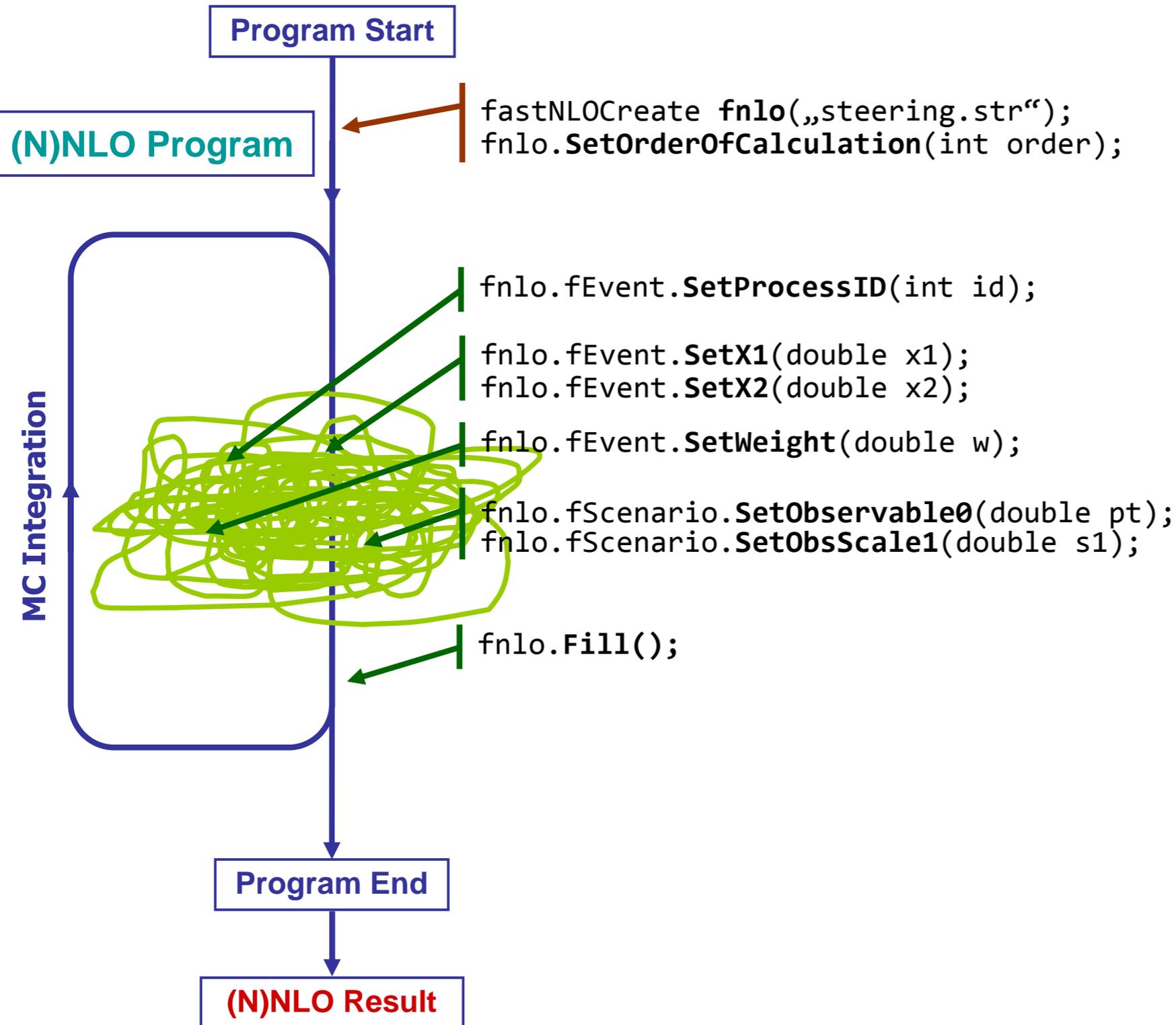
Application procedure of new 'fastNLO Toolkit'



Application procedure of new 'fastNLO Toolkit'



Application procedure of new 'fastNLO Toolkit'



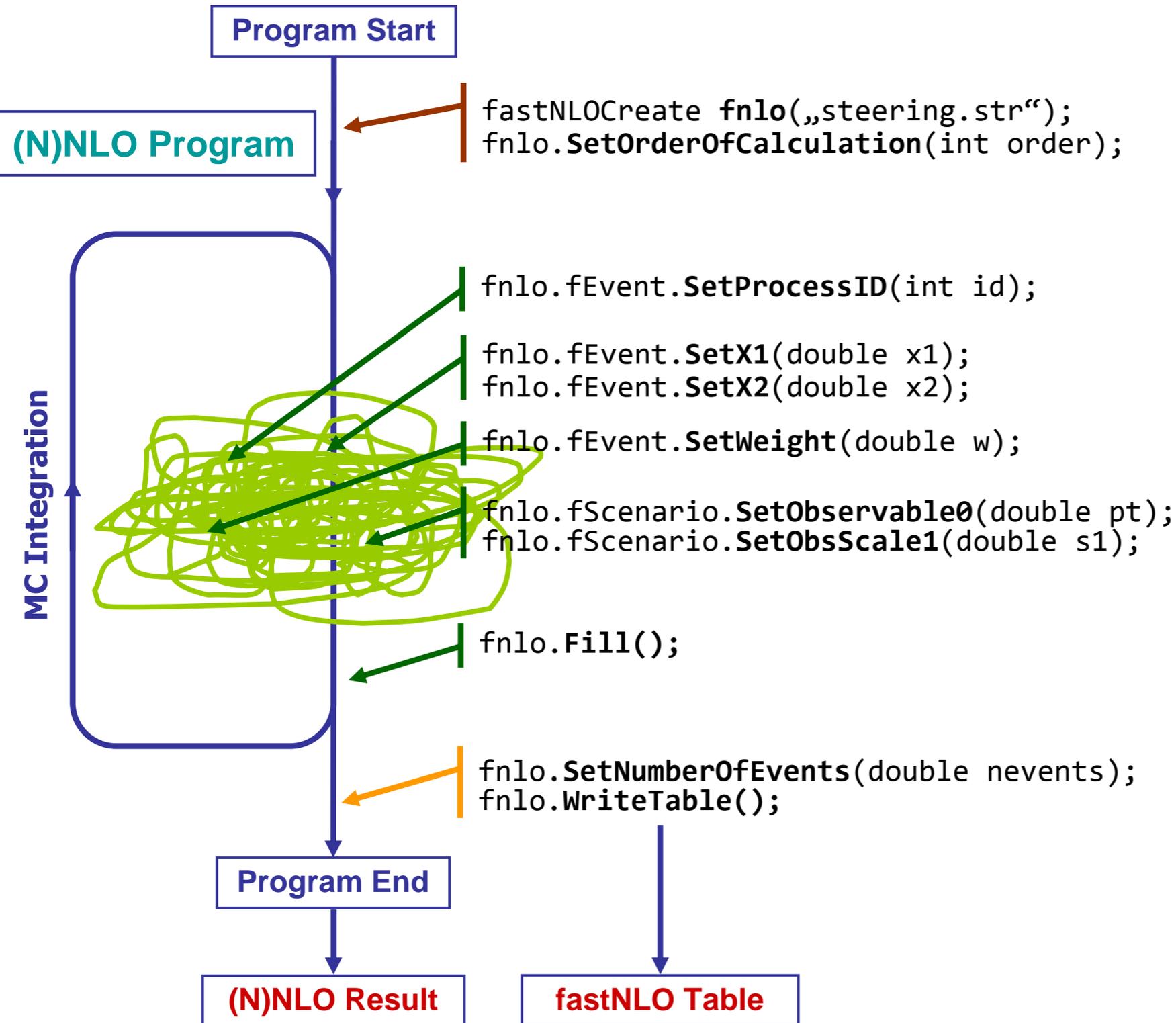
Initialize fastNLO class(es)

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

- Order does not matter
- Many other convenient implementations possible

Pass all information to fastNLO

Application procedure of new 'fastNLO Toolkit'



Initialize fastNLO class(es)

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

- Order does not matter
- Many other convenient implementations possible

Pass all information to fastNLO

Set normalization of the MC integration and write table

Minimum implementation: 11 lines of code

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

New developments for fastNLO toolkit

1) Creation of fastNLO tables

- One stand-alone c++ library
 - No third party packages needed
 - Optimally: Only 11 lines of code necessary
 - Depending on (N)NLO program: many implementations possible
- Parameters are specified in steering card
 - Binning (also double- or triple- differentially)
- Performance optimized
 - Caching of interpolation values
 - Automatic optimization of grids to phase space
- PDF parton combinations for different subprocesses
 - Specified in steering
 - Stored in table

2) fastNLO table format

- Further contributions are foreseen
 - EW corrections, etc...
- PDF combinations are stored in table
- Storage of uncertainties soon available
- QEDPDFs or p-Pb collisions also foreseen

3) Evaluating fastNLO tables

- New interface in PYTHON
- Many interfaces to PDF and α_s routines
 - LHAPDF5, LHAPDF6, Hoppet, QCDNUM, ALPHAS, CRunDec, ...
- Improved speed for flexible-scale tables
- Caching of PDFs and α_s values for (even) faster re-evaluation

```
// FastNLO example code in c++ for reading CMS incl.  
// jets (PRL 107 (2011) 132001) with CT10 PDF set
```

```
fastNLOLHAPDF fnlo("fnl1014.tab","CT10.LHgrid",0);  
fnlo.PrintCrossSections();
```

fastNLO enables fast re-evaluation of perturbative calculations

- Convenient for scale or PDF studies (e.g. uncertainties)
- Mandatory tool for phenomenological analysis (e.g. α_s or PDF fits)

Scale-independent concept successfully applied in NNLO

fastNLO is applicable to a wide range of processes and corrections

New tool: 'fastNLO toolkit'

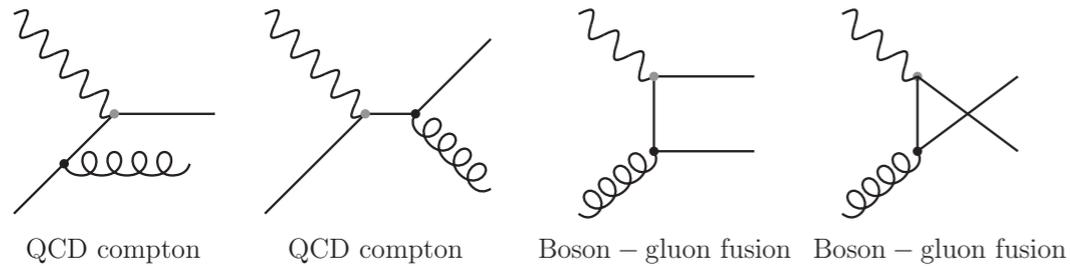
- Facilitates creation of tables and interface to other (N)NLO programs
- Very flexible code:
Only few modifications in (N)NLO programs are needed (O(11) lines of code)

Code and manual is released soon after conference

- Python interface available for reading tables
- pre-release version of 'fastNLO toolkit' is available on request
- more information: <http://fastnlo.hepforge.org>

Application of flexible scale concept

Inclusive jet production in DIS



Two scales are stored in table

- Q^2
- p_T of the jet

Any function of the two can be used as scale

Renormalization and factorization scale can be varied separately

Choose for scale study

- $\mu_r^2 = Q^2 + p_T^2$
- $\mu_f^2 = Q^2$
- Color code shows 5% change in cross section w.r.t. to scale factor of 1

Other scale choices also shown without scale factors

