PDFFlow: hardware accelerating parton density access

Marco Rossi[1,2]

in collaboration with S. Carrazza[1] and J.C. Martinez[1]

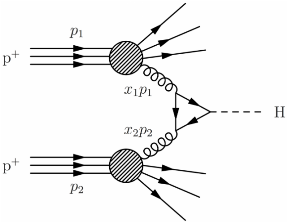
[1]University of Milan [2]Openlab-CERN

40th International Conference on High Energy Physics
Prague - 2020

Proton-proton interactions at LHC

QCD factorization:

- hard scattering $gg \to H$ (pQCD)
- long-distance universal functions: partonic distribution functions (PDFs)



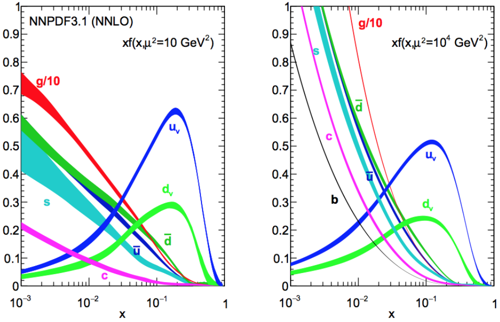
Higgs production in gluon gluon fusion,
leading diagram at LHC

Master formula:

$$\sigma = \sum_{a,b} \int_0^1 dx_1 dx_2\, f_a(x_1, Q^2)\, f_b(x_2, Q^2)\, \hat{\sigma}_{a,b}(x_1, x_2, Q^2)$$

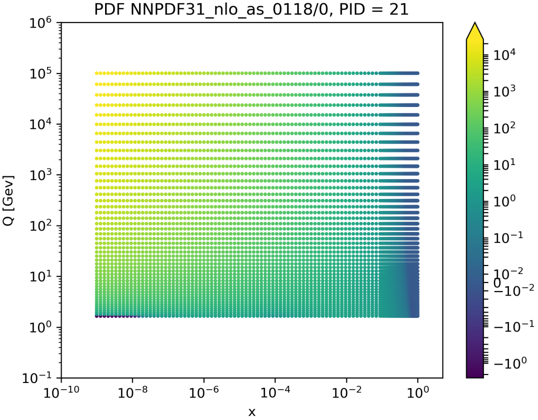
Parton distribution functions

- PDF $f_a(x, Q^2)$ evolution with Q^2 is driven by the DGLAP equation
- PDFs are essential for theoretical calculations
- Modern PDFs are accompanied by uncertainty bands that could slow down the convolution evaluation



PDF grid format

- LHAPDF6 provides official pdf sets
- A PDF set contains versions with grids for all flavors
- Only some points are measured: interpolation needed
- Interpolation in grid range: $(x, Q) \in [10^{-9}, 1] \times [1.65, 10^5]$
- Need to extrapolate outside grid range

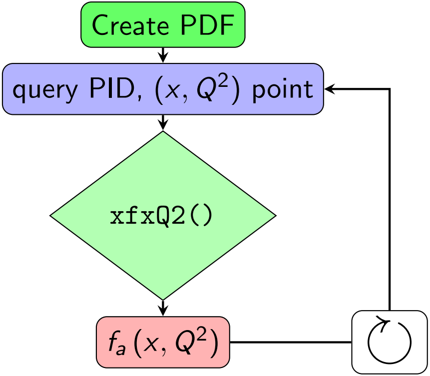


Example: PDF points from gluon NNPDF31_nlo_as_0118/0

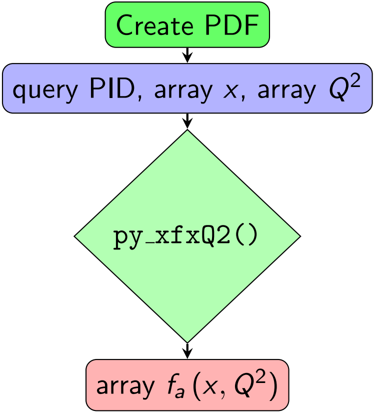
Parallel vs Sequential Queries

Mimic the LHAPDF interpolation methods, parallelize them

LHAPDF6 algorithm:



PDFFlow algorithm:



! Query points are independent

✓ PDFFlow benefits from hardware acceleration (multithreading CPU, GPU)

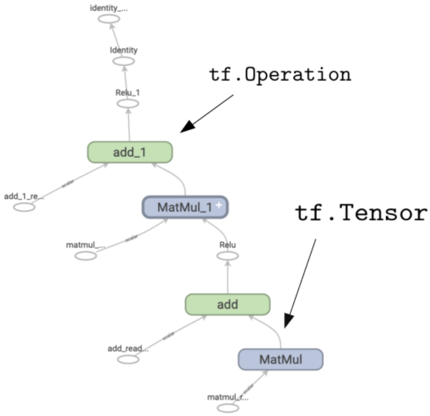
TensorFlow library

- End-to-end open source platform for machine learning (by Google)
- Rich python API library: `tf v2.x` compatibility
- Automatic multi-threading CPU and GPU management
- No need to go through specific GPU code (CUDA, OpenCL)
- *Eager* vs *graph* modes

TensorFlow Graph Mode

Wrapping usual eager code with `tf.function`:

- ✓ graph optimization (faster computation)
- ✓ run faster (high parallelization, constant folding, . . .)
- ! Warning: retracing (graph building time overhead)

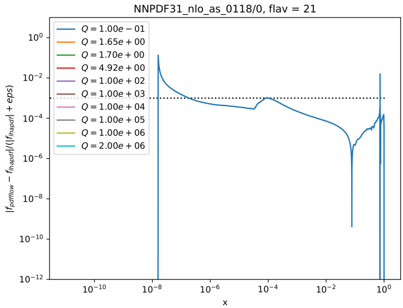


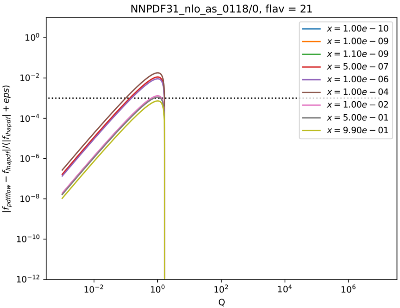
Accuracy benchmark

Absolute relative difference of `PDFFlow` against `lhapdf`:

$\frac{|f_{lhapdf} - f_{pdfflow}|}{|f_{lhapdf}| + \epsilon}$, where regulator ϵ avoids division by 0.

Acceptable error threshold is 10^{-3} according to LHAPDF6 paper.

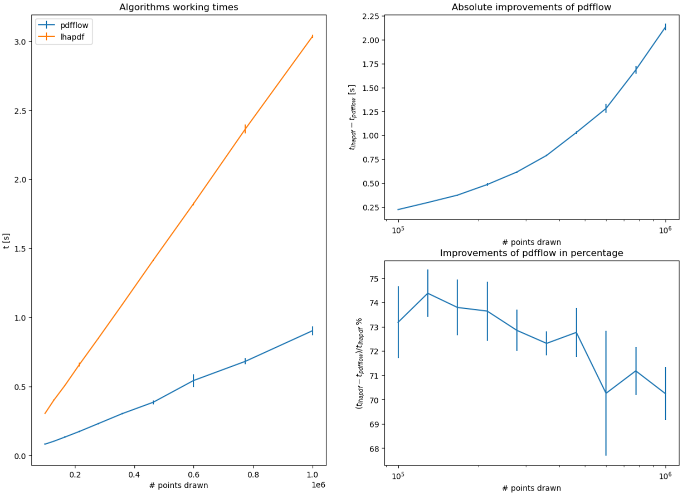




Some values above threshold for low Q extrapolation algorithm.

Performance benchmark - CPU Intel i9 9980xe, 36 cores

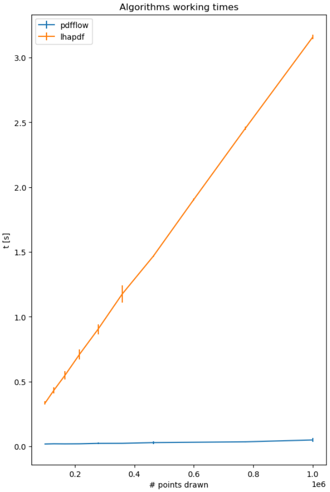
PDF set: NNPDF31_nlo_as_0118, PDF ID: 0. Parton: gluon

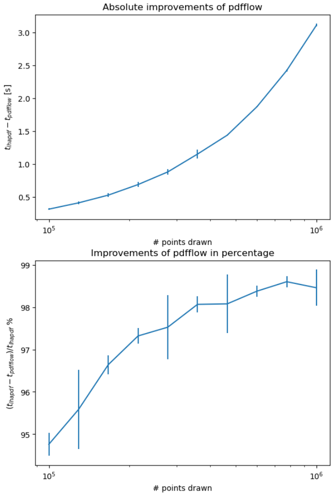


✓ $\sim 72\%$ OF RELATIVE IMPROVEMENT

Performance benchmark - GPU Nvidia Titan V

PDF set: NNPDF31_nlo_as_0118, PDF ID: 0. Parton: gluon





✓ $\sim$ 98% OF BEST RELATIVE IMPROVEMENT

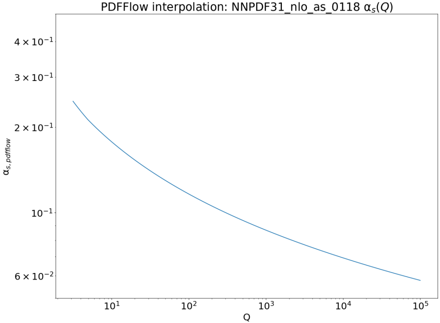
Strong running coupling

The strong coupling $\alpha_s(Q)$ evolution is driven by renormalization group equation

Modern PDF sets come with a grid for α_s points in Q space

Mimic the `PDFFlow` algorithm in 1 dimension

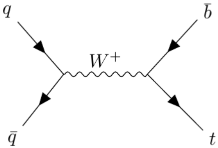
✓ Exact matching between `PDFFlow` and `LHAPDF` interpolations



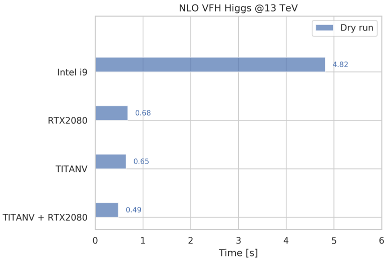
Physical example - Single top production at LO

Combine `PDFFlow` and `VegasFlow` (MC integrator, 10.1016/j.cpc.2020.107376)

Speed comparison CPU-GPU for `PDFFlow` + `VegasFlow`



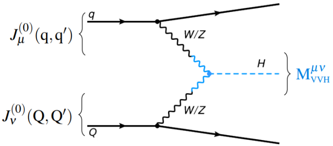
Born diagram for $q\bar{q} \to \bar{b}t$



Single top dry run

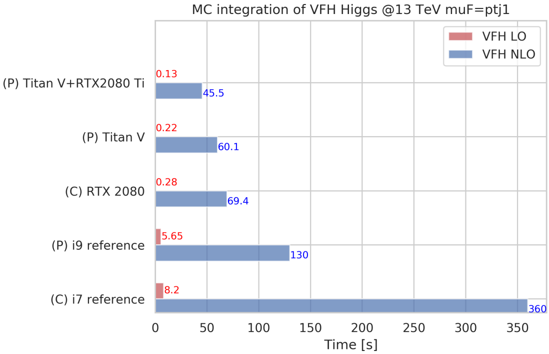
✓ Speed-up range: $[7.0, 9.9]x$

Physical example - VBF Higgs production at NLO



Born diagram for $qQ \to qQH$

hep-ph/arXiv:1807.07908



(C) consumer-grade
(P) professional-grade hardware

- ✓ Best speed-up at LO: 63x(C), 43x(P)
- ✓ Best speed-up at NLO: 7.9x(C), 2.9x(P)

CPU implementation: `LHAPDF` + Fortran code
GPU implementation: `PDFFlow` + `VegasFlow`

Summary

We presented `PDFFlow`, the first GPU port for PDF interpolation:

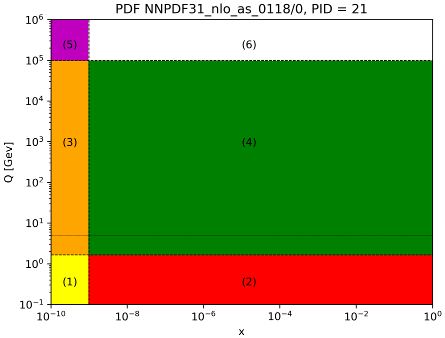
- benchmark against LHADPF6 for performance and accuracy, achieving a notable speedup via parallelized and `TensorFlow` optimized algorithm
- implemented α_s strong running coupling interpolation
- provided application examples: single-top at LO, VFH at NLO
- `VegasFlow-PDFFlow` outperforms standard CPU algorithm for either consumer and professional grade GPU

Code beta release:

public and available at https://github.com/N3PDF/pdfflow
DOI is available here

Thanks!

Backup slides - PDF interpolation zones



(4) Inside grid: bicubic interpolation

(3)-(6) Low x and high Q^2 regions: (log-)linear extrapolation from the two nearest grid knots

(2) Low Q^2 region: anomalous dimension interpolation between $\gamma(Q_{min})$ and $1 \rightarrow xf(x, Q^2) = xf(x, Q_{min})\left(Q^2/Q^2_{min}\right)^{\gamma(Q^2_{min})}$

(1) Low x, Low Q^2 region: extrapolation algorithm is $(2) + (1)$

(5) High x, High Q^2 region: extrapolation algorithm is $(3) + (6)$

Backup slides - α_s interpolation zones

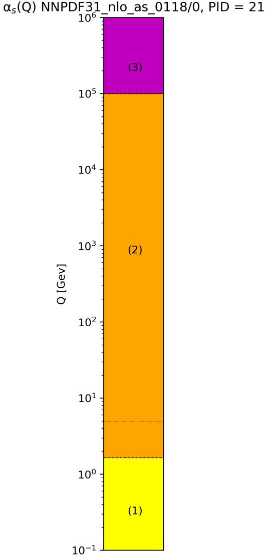
(1) Low Q region:

$$\alpha_s(Q) = \alpha_s(Q_{min})\,(Q^2/Q^2_{min})^{\left.\frac{\partial \log \alpha_s}{\partial Q^2}\right|_{Q^2=Q^2_{min}}}$$

for $Q < Q_{min}$

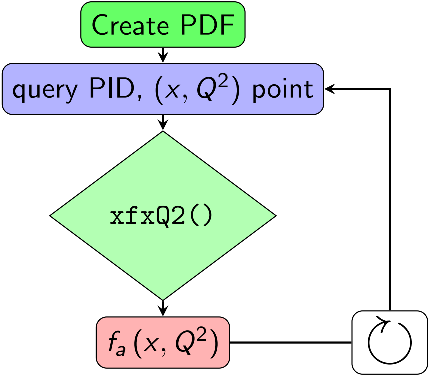
(2) Inside α_s grid: cubic interpolation

(3) High Q region: $\alpha_s(Q) = \alpha_s(Q_{max})$ for $Q > Q_{max}$



Backup slides - Usage example

LHAPDF6 algorithm:

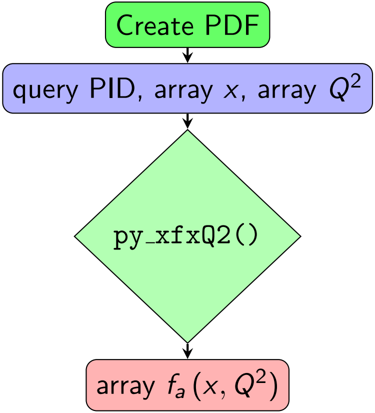


Code snippet:

```
>>> import numpy as np

>>> from lhapdf import mkPDF

>>> p = mkPDF(pdfname, DIRNAME)

>>> xs = np.logspace(-10,0,100000)

>>> q2s = np.logspace(0,6,100000)

>>> for x in xs:
>>>   for q2 in q2s:
>>>     p.xfxQ2(21, x, q2)
```

Backup slides - Usage example

PDFFlow algorithm:



Code snippet:

```
>>> import numpy as np

>>> from pdfflow.pflow import mkPDF

>>> p = mkPDF(pdfname, DIRNAME)

# first build the graph with the least
# number of points needed to save time
# later on the first interpolation
>>> p.trace()

>>> xs = np.logspace(-10,0,1000000)

>>> q2s = np.logspace(0,6,1000000)**2

>>> p.py_xfxQ2(21,xs,q2s)
```