PDFxTMDLib: A High-Performance C++ Library for Collinear and Transverse Momentum Dependent Parton Distribution Functions

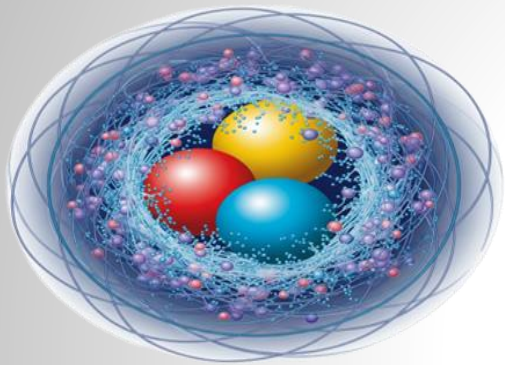
Ramin Kord Valeshabadi & Somayeh Rezaie

Presented by Ramin Kord Valeshabadi

30th July 2025

➢ What are Partons and PDFs?

Partons:



Protons and neutrons are not fundamental. They are made of fundamental particles called **partons**.

Quarks (e.g., up, down)

Gluons (the particles that "glue" quarks together)

- **PDFs:**

A **Parton Distribution Function (PDF)** tells us the probability of finding a specific parton inside a proton at a given energy.

It's represented as: $f_i\ (x, \mu^2)$

- x: How much **momentum** does it carry? (A fraction from 0 to 1)
- μ^2: The energy scale of the interaction (factorization scale)

➢ Why are PDFs important?

- **Predictive Power**: PDFs are essential for calculating the cross-sections of particle interactions at high-energy colliders like the Large Hadron Collider (LHC). The cross-section is a measure of the probability of a particular interaction occurring. Generally, factorization formula allows us to calculate cross section as a convolution of PDFs and partonic cross sections, e.g.:

$$\sigma = \sum_{i,j \in q,g} \int \frac{dx_1}{x_1} \frac{dx_2}{x_2} f_i(x_1, \mu_1^2)\, f_j(x_2, \mu_2^2)\, \hat{\sigma}_{ij}$$

- **Understanding Hadron Structure**: PDFs provide fundamental insights into the internal structure of protons and neutrons, revealing how the momentum is distributed among their constituent quarks and gluons.
- **Universality**: PDFs are universal, meaning they are independent of the specific process in which the proton is involved. Once determined from one set of experiments, they can be used to make predictions for others.

➢ How are PDFs Determined?

- **Non-Perturbative Nature**: PDFs describe the long-distance structure of the proton and **cannot be easily calculated from first principles**.
- **Global Fits**: Instead, they must be extracted from experimental data. This is achieved through a complex statistical procedure known as a **global fit**.
- **Process**: The process starts by assuming a flexible functional form (a parameterization) for the PDFs at an initial low energy scale, μ_0^2. These initial PDFs are then evolved to higher scales ($\mu^2 > \mu_0^2$) using the DGLAP equations. The parameters of the initial function are then adjusted ("fitted") by comparing the resulting theoretical cross-section calculations with a vast amount of data from many different high-energy experiments (from colliders like HERA, the Tevatron, and the LHC) until the best possible agreement is achieved.

➢ DGLAP Evolution Equations

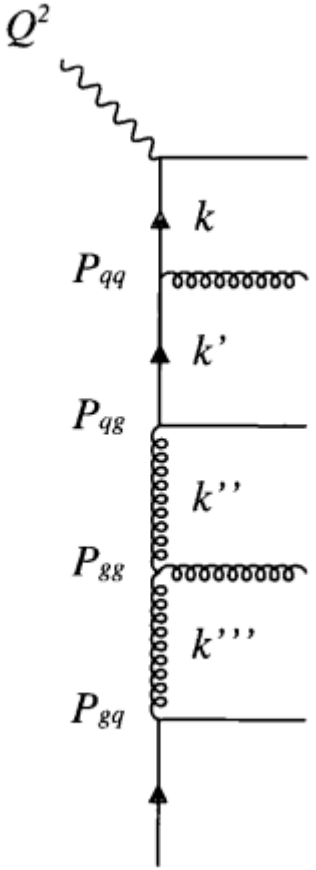
- DGLAP evolution equations describe how Partor Distribution Functions (PDFs) change as the energy scale (μ^2) of the interaction varies. They are the cornerstone of perturbative QCD for calculating hadron structure :
- The DGLAP formalism is derived under the assumption of **strong ordering** in the virtuality (μ^2) of the emitted partons, which implies that transverse momenta can be neglected. This leads to a corresponding ordering in the longitudinal momentum fraction, x:

  virtuality ordering:

  $$\mu_0^2 \ll \mu_1^2 \ldots \ll \mu^2$$

  Momentum fraction ordering:

  $$x_0 > x_1 \ldots > x$$



➢ TMD Evolution: Beyond DGLAP

- While the DGLAP framework is highly successful, it is based on the approximation of strong ordering in virtuality (μ^2), which is valid in a large kinematic region. However, to describe phenomena where the parton's transverse momentum (k_t^2) is significant, especially at small values of the momentum fraction x, we need more general evolution equations.

➢ From Collinear to Transverse Momentum:

- Beyond Collinearity: While the DGLAP framework assumes partons are collinear with the parent hadron, a more complete picture requires considering their intrinsic transverse momentum,k_t.

- k_t-factorization: This is a theoretical framework that incorporates the transverse momentum of partons. The hadronic cross-section is expressed as a convolution of the partonic cross-section, and **Transverse Momentum Dependent (TMD) PDFs**:

$$\sigma = \sum_{i,j \in q,g} \int \frac{dx_1}{x_1} \frac{dx_2}{x_2} \frac{dk_{1,t}^2}{k_{1,t}^2} \frac{dk_{2,t}^2}{k_{2,t}^2} f_i\left(x_1, k_{1,t}^2, \mu_1^2\right) f_j\left(x_2, k_{2,t}^2, \mu_2^2\right) \hat{\sigma}^*_{ij}$$

➢ BFKL Evolution

- The Balitsky-Fadin-Kuraev-Lipatov (BFKL) equation is working for the small-x regime, where gluon densities are high.
- **Kinematic Assumption**: In contrast to DGLAP, BFKL evolution is based on strong ordering in the longitudinal momentum fraction, x, with no ordering in virtuality. This means the transverse momenta of emitted partons can be comparable to their virtuality.

$$x_0 \gg x_1 \gg \ \ldots \gg x$$
$$k_t \approx \mu$$

- BFKL is limited to gluon, and obtaining quark TMDs are not possible.

➢ CCFM Evolution

- The Catani-Ciafaloni-Fiorani-Marchesini (CCFM) evolution equation provides a unified framework that bridges the gap between DGLAP and BFKL.
- **Kinematic Assumption**: **Angular Ordering**. The key innovation of CCFM is that the evolution is governed by the angle of gluon emissions. The emissions are ordered in a variable that is related to the emission angle with respect to the incoming parton.
- **Interpolating Behavior**: This angular ordering scheme naturally reproduces the other evolution equations in their respective limits.
- For emissions at **large angles**, it behaves like **BFKL** evolution.
- For emissions at **small angles**, it reduces to **DGLAP** evolution.
- CCFM evolution equation is limited to gluon similar to BFKL.

➢ PB approach

- The Parton Branching (PB) method is a powerful and successful technique for constructing Transverse Momentum Dependent (TMD) parton distributions for both quarks and gluons. It provides a direct, dynamical link between the well-understood collinear PDFs (cPDFs) and the more complex TMDs.
- The PB approach simulates the evolution of partons from an initial low scale to a higher factorization scale using a Monte Carlo method, and DGLAP evolution equations.
- It assumes the Gaussian transverse momentum dependency at the initial scales, and then evolves partons along the evolution ladder.
- It allows to calculate TMDs for both quark and gluon.

➢ How cPDFs and TMDs are obtained for phenomenology?

- Generally, cPDFs and TMDs are provided in the form of grid files by different groups, where interpolation-based libraries are used to calculate them.
- The LHAPDF library is widely used to access cPDFs for calculating cross sections within the collinear factorization framework.
- TMDLib is commonly employed for TMDs in cross section calculations within the *kt*-factorization framework.
- These libraries primarily facilitate interpolation-based access, enabling efficient retrieval of cPDFs via two dimensional interpolations and TMDs via three-dimensional interpolations.

➢ Problems with LHAPDFs, and TMDLib

- Their core design, based on fixed-dimensional interpolation users are confined to the libraries' built-in algorithms, with no straightforward way to implement custom interpolation or extrapolation methods.
- lacks the extensibility required for modern, and future phenomenological studies involving higher-order distributions like Double Parton Distribution Functions (DPDFs), etc.
- Good library should not be limited to certain specific dimensionality.
- TMDLib, and LHAPDF code bases convinced me we need a new library. Their APIs are not easy to use and lead you to different bugs in your code.
- With these tools we have two different APIs for the same entity, i.e. PDFs.

➢ TMDLib Limitations

- TMDLib currently lacks a standard for the structure of the TMD set grids, or its info file.
- TMDLib limiting you from QCD coupling, and uncertainty calculation. Even calculating uncertainty with LHAPDF is difficult.
- Lacks of support of QCD coupling, leads to Monte Carlo event generators like KATIE use LHAPDF QCD coupling for their k_t-factorization calculation!
- TMDLib code base is comprised of different TMD groups code under the umbrella of a unified API. Hence, there are many different implementations to calculate TMDs of different groups which leads to lurking more bugs in your TMD calculation.

➢ PDFxTMDLib: Key Features & Innovations (1)

1. Performance & Extensibility:
   - **Unified Framework:** Provides a single, high-performance interface for both collinear PDFs (cPDFs) and Transverse Momentum Dependent PDFs (TMDs).
   - **Performance:** Modern C++ design makes it generally faster than traditional libraries like LHAPDF and TMDLib.
   - **Future-Proof Design:** The architecture is built for extensibility, paving the way to support higher-order distributions like Double Parton Distributions (DPDFs) and Double TMDs (DTMDs).
2. Novel Capabilities:
   - **Integrated Uncertainty Calculations:** For the first time in a TMD framework, PDFxTMD allows for the calculation of PDF uncertainties and correlations directly from a PDF set.
   - **QCD Coupling(α_s):** Includes built-in functionality to compute the QCD running coupling, a crucial component for precision physics that was absent in previous TMD libraries.

➢ PDFxTMDLib: Key Features & Innovations (2)

3. Portability & Ease of Use:
   - **Standalone & Portable:** Unlike other libraries that depend on many third-party tools, PDFxTMDLib is a standalone package, simplifying installation and use.
   - **Cross-Platform & Multi-Language:** It is fully supported on both Linux, Windows, and Mac OS with a native C++ interface and a user-friendly Python wrapper.

4. Flexible Data Handling:
   - **New TMD Grid Format:** Addresses the lack of a standard for TMD grids by introducing a new format that extends the familiar lhagrid1 to three dimensions
   - **Support for Custom Formats:** The modular design allows PDFxTMDLib to easily read and handle non-standard or user-defined grid formats.

➢ How PDFxTMDLib is Fast? (1)

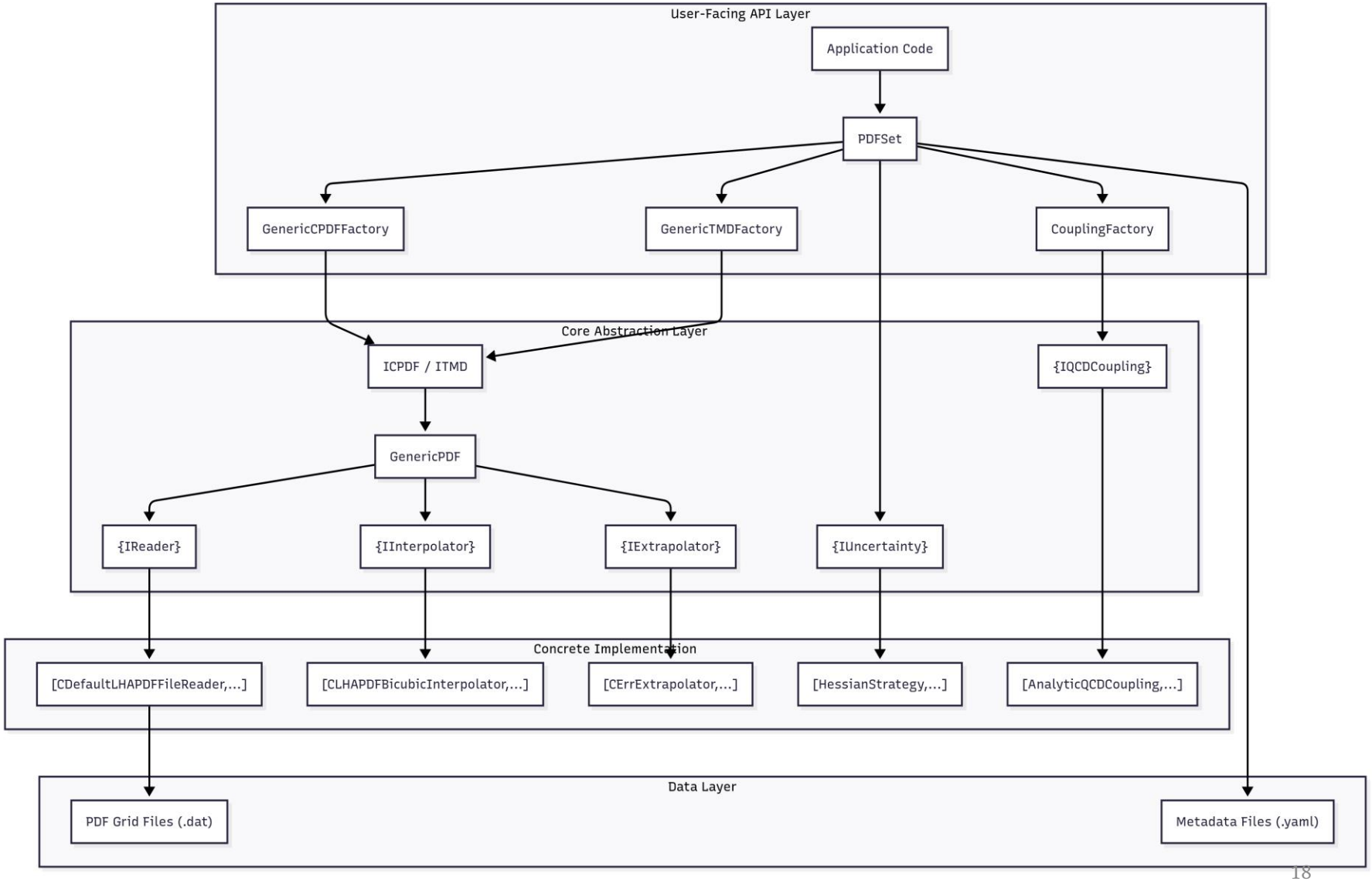
1. Compile-Time Polymorphism with CRTP:
   - **The Problem with Virtual Functions:** Traditional object-oriented designs use virtual functions for polymorphism. This requires a vtable lookup at runtime to determine which function to call, introducing an overhead that can be significant in performance-critical code like PDF evaluation.
   - **The CRTP Solution:** PDFxTMDLib uses the **Curiously Recurring Template Pattern (CRTP)**. This is a form of compile-time (or "static") polymorphism.
   - **class ConcreteInterpolator** :

     public BaseInterpolator<ConcreteInterpolator> { ... };
   - **The Benefit: No Runtime Overhead.** With CRTP, the compiler knows the exact function to call at compile time. This eliminates the vtable lookup and enables powerful optimizations like **function inlining**, resulting in much faster execution.

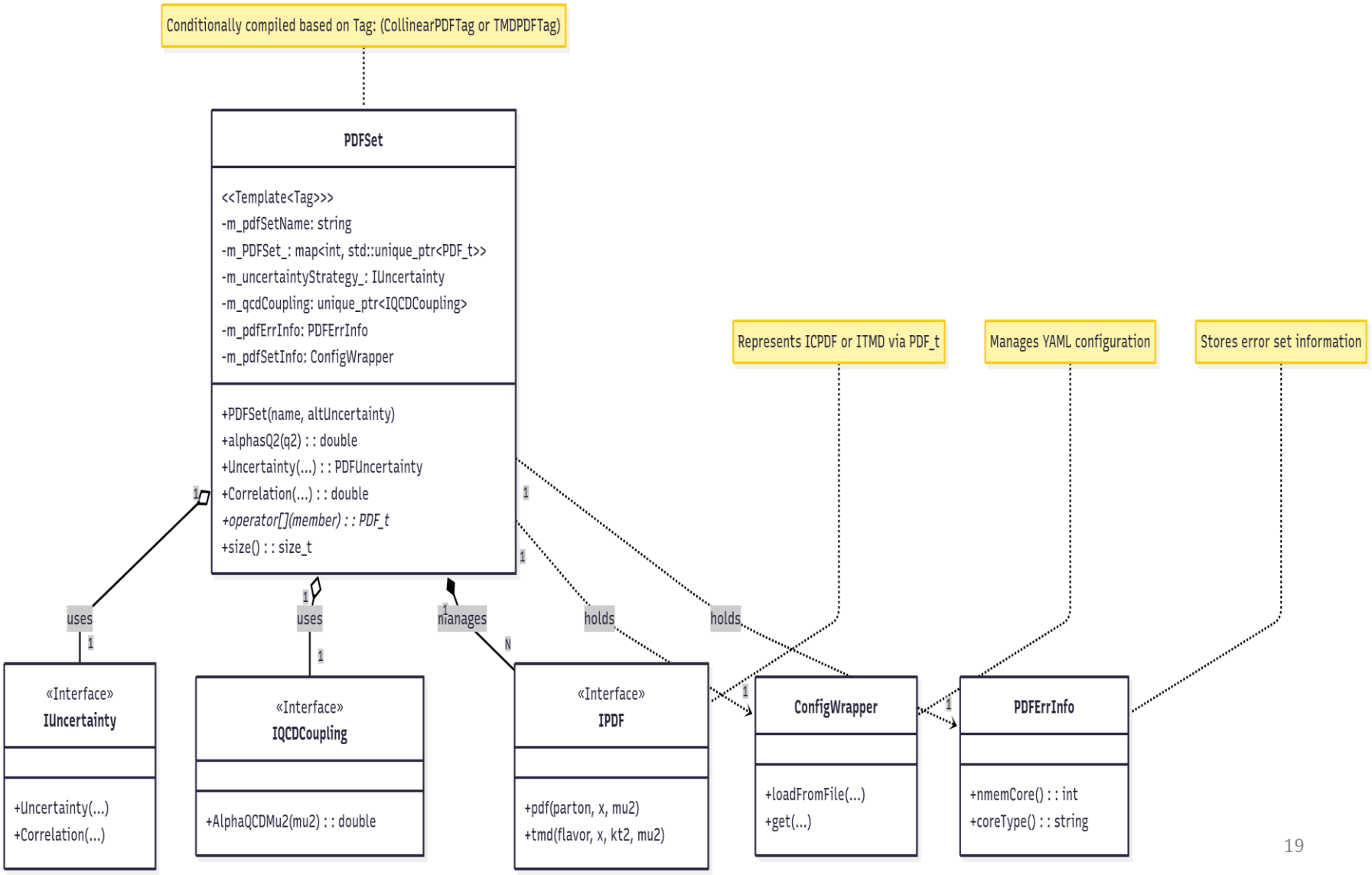
➢ How PDFxTMDLib is Fast? (2)

2. **The Challenge**: While CRTP is fast, we still need a way to handle different PDF types. Because PDF sets info are read from file (runtime).
   - **The Solution**: **Type Erasure.** Instead of using a standard inheritance hierarchy with virtual functions, PDFxTMDLib uses a technique called **type erasure**. It manually implements its own function dispatch mechanism.
   - **The Benefit**: **Control and Performance.** This creates a wrapper that can hold any object that conforms to a specific concept. Therefore, users does not need to directly inherit from the base class, and only needs to follow its APIs.

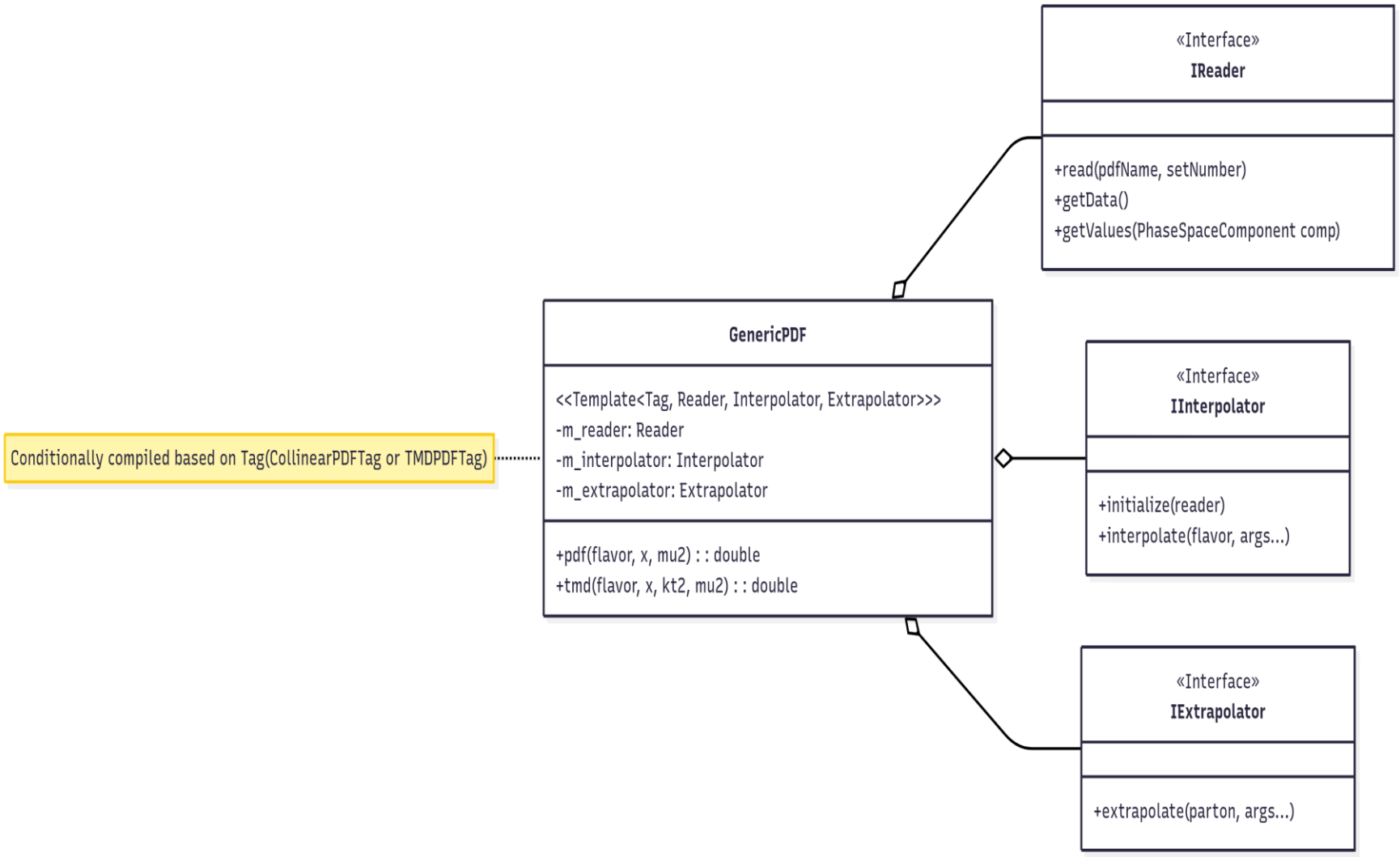
➢ Bird-eye view of the architecture



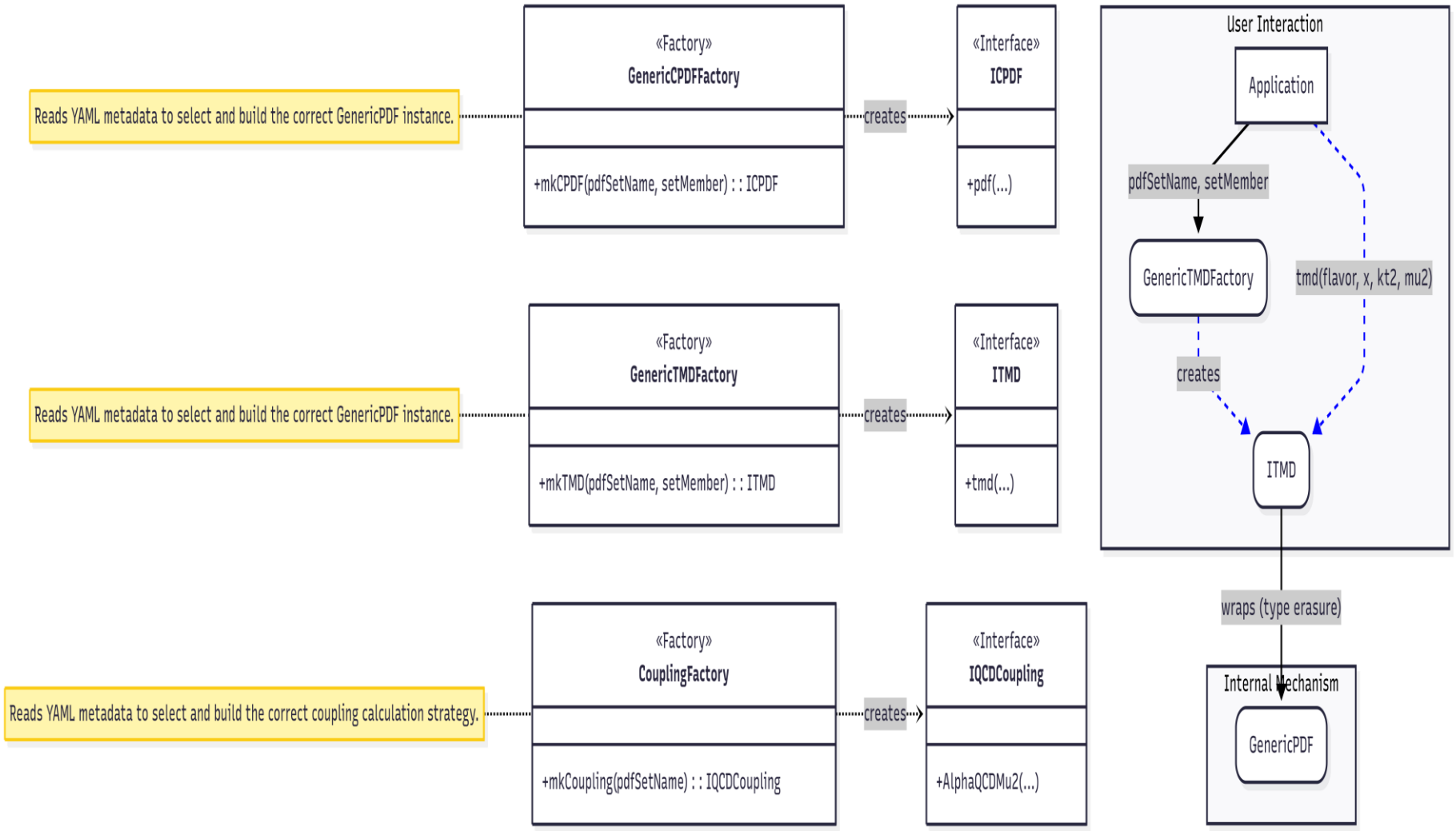
➢ PDFSet Class



➢ GenericPDF class



➢ Factory classes



➢ Filesystem Hierarchy & Configuration

- For PDFxTMD to find your datasets, they must be organized in a specific directory structure:

  PDFxTMD_PATH/<setname>/<setname>_<nnnn>.dat

  PDFxTMD_PATH/<setname>/<setname>.info

  - <setname> is the name of the PDF set (e.g., CT18NLO)
  - <nnnn>is the four-digit member ID (e.g. 0000 for the central member).
  - The .info fiile contains the YAML-formatted metadata for the set.

- Configuring Custom Search Paths:
  - You can tell PDFxTMDLib where to find your PDF sets by editing the config.yaml file.
  - File Location:
    - Windows:C:\ProgramData\PDFxTMDLib\ config.yaml
    - **Linux/Unix:** ~/.PDFxTMDLib/config.yaml
  - Example config.yaml:
    - paths:

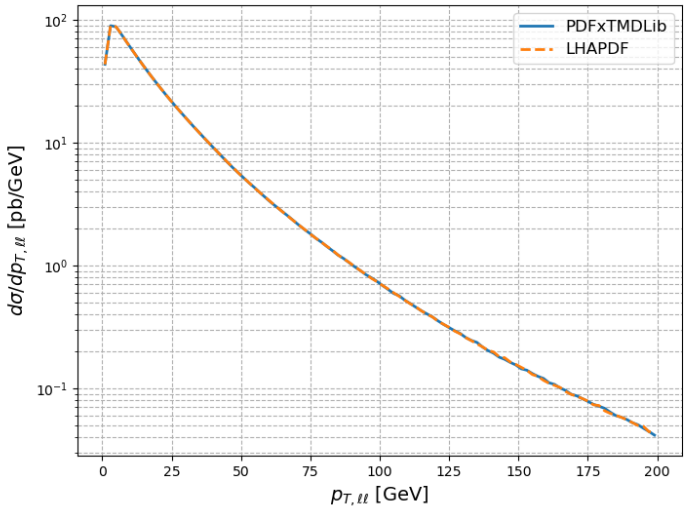
      -/path/to/my/pdf_sets_1

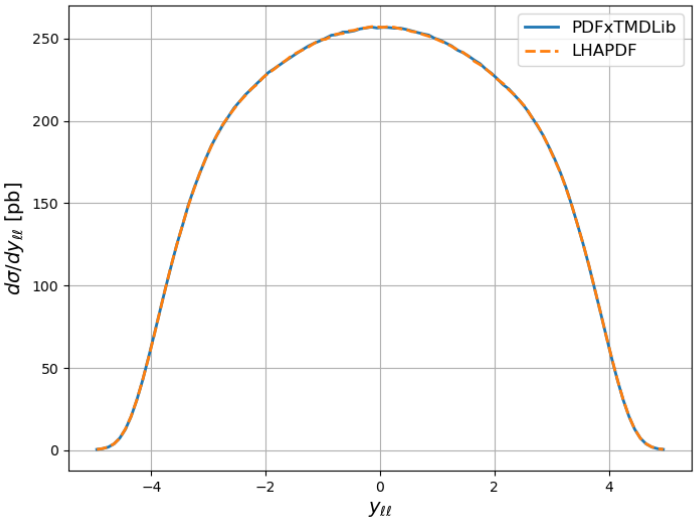
      - /another/path/to/pdf_sets_2
- Search Path Logic:
  1. The current directory where the program is running.
  2. The system-wide installation directory (e.g., /usr/local/share/PDFxTMDLib).
  3. Any custom paths defined in your config.yaml file.

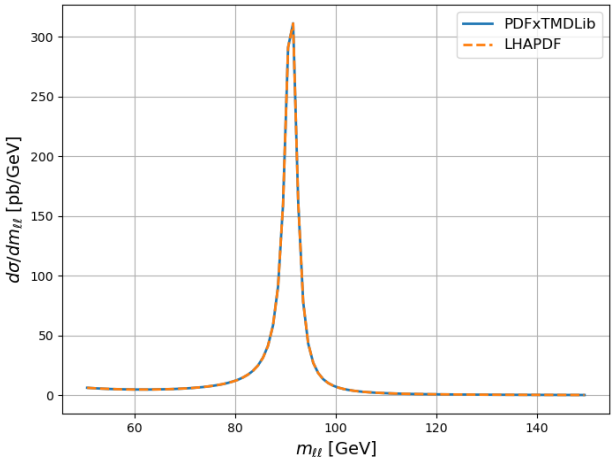
➢ Validation I: Drell-Yan Simulation in PYTHIA

- To validate the numerical accuracy and performance of PDFxTMDLib, we use the Drell-Yan process, a key benchmark for both collinear and TMD physics.
- **Methodology**
  - Simulation: Proton-proton collisions at $\sqrt{s}$ = 13 TeV, simulating $Z \rightarrow \mu^+\mu^-$ with 50 million events.
  - Toolchain: An extended version of the PYTHIA 8 Monte Carlo event generator.
  - Comparison: Results are compared against LHAPDF using the MSHT20nlo_as120 PDF set.
- **Performance Results**
  - Execution Time (50M events):
    - LHAPDF: 171,935 seconds
    - PDFxTMDLib: 162,263 seconds
  - Improvement: A ~5.6% reduction in total execution time.
- **Accuracy Results**
  - Total Cross Section: Measured as 1.695×10^{-6} mb for both libraries.
  - Differential Cross Sections: Excellent agreement across all kinematic variables.

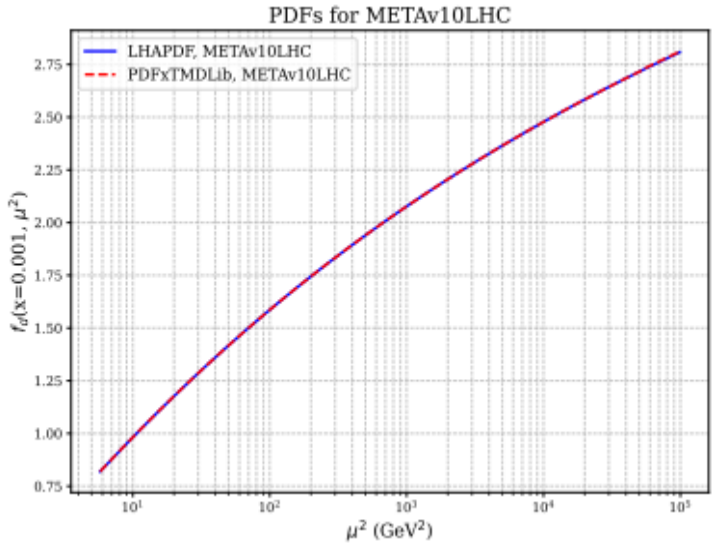
➢ Validation II: Differential Cross-Section

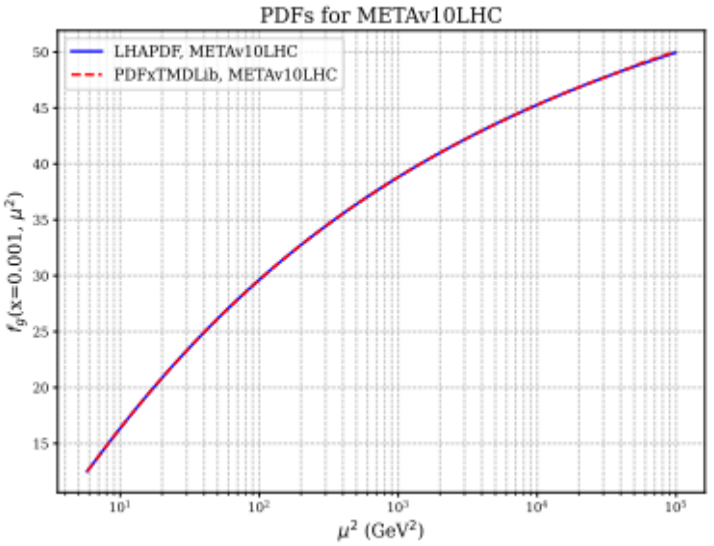


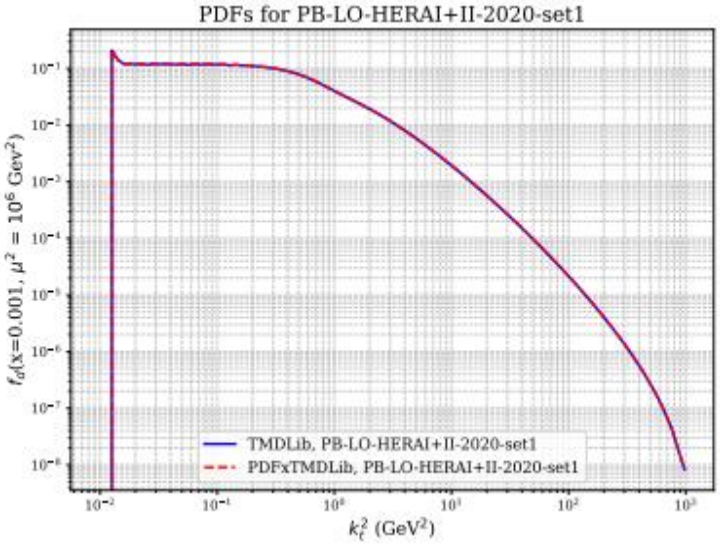


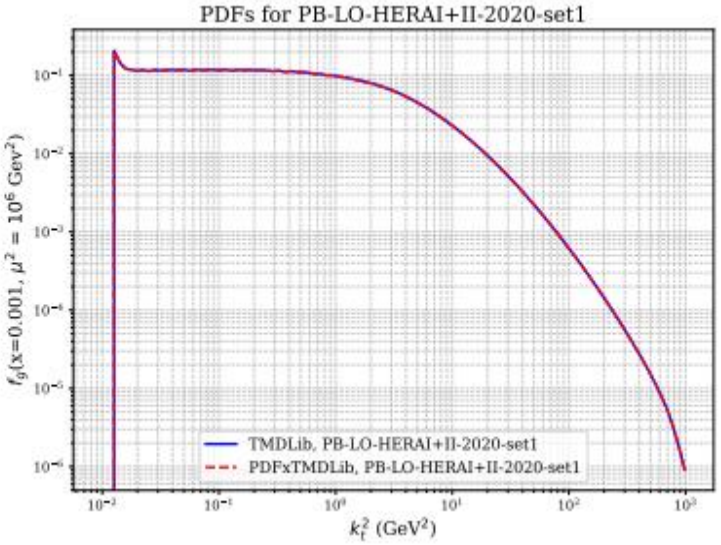


➢ Validation III: TMD & cPDF comparisons









➢ Installation & Integration Methods

- Building: Prerequisites: C++17 compiler, CMake 3.14+
  - Build process:
  - mkdir build && cd build
  - cmake -DCMAKE_BUILD_TYPE=Release ..
  - cmake --build .
  - cmake –install .
- Integration Methods:
  - Cmake:
    - find_package (PDFxTMDLib REQUIRED)
    - target_link_libraries (your-target-name PDFxTMD::PDFxTMDLib)
  - *Direct Compilation:*
    - Linux: g++ -std =c ++17 your_source. cpp - lPDFxTMDLib -o your_executable
    - Windows: cl your_sur_source.cpp /std:c++17 PDFxTMDLib . lib

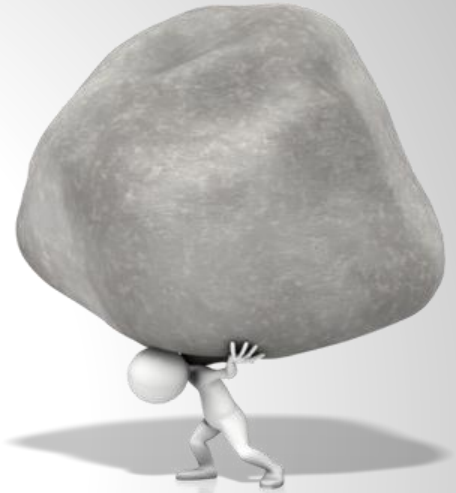
➢ Conclusion

- PDFxTMDLib: High-performance, extensible C++ library
- Unified handling of cPDFs and TMDs with new file format
- Validated accuracy and improved performance
- Future work: More PDF/TMD sets, algorithmic optimizations

➢ Questions & Answers

- Thank you for your attention!
- Contact: ramin.kord@ipm.ir, raminkord92@gmil.com
- GitHub: https://github.com/Raminkord92/PDFxTMD
- PDFxTMD official website: **www.pdfxtmdlib.org**

➢ Demo:



https://colab.research.google.com/drive/1c0lJt0XXNKJxVRzYJcY0Pe4IwVGLUIwy#scrollTo=YsVAvxKIQ2B6

https://colab.research.google.com/drive/1C_h9oGJJzt5h3m-h6o3lAJ5dl-AVv0j3#scrollTo=p7rEfvtq1eUi