PARTONS project to study Generalized Partons Distributions

Paweł Sznajder (on behalf of PARTONS Collaboration)

Institut de Physique Nucléaire, Orsay
National Centre for Nuclear Research, Warsaw

Various Faces of QCD 2, Świerk, 8-9 October 2016

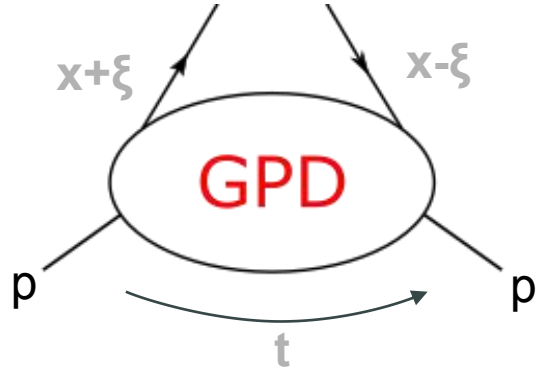
OUTLINE

- Motivation
- Introduction to PARTONS project
- Content and performance
- Summary

GPDs (Generalized Parton Distributions)

- 3D functions describing partonic structure of nucleon
- Each one defined for specific parton and specific helicity configuration
- Studied in various experimental channels
- pQCD involved

handbag diagram:



chiral-even GPDs:

$H^{g,q}(x, \xi, t)$ $E^{g,q}(x, \xi, t)$ *for sum over parton helicities*

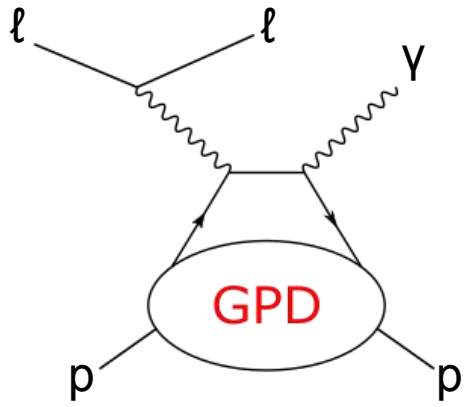
$\tilde{H}^{g,q}(x, \xi, t)$ $\tilde{E}^{g,q}(x, \xi, t)$ *for difference over parton helicities*

nucleon helicity conserved *nucleon helicity changed*

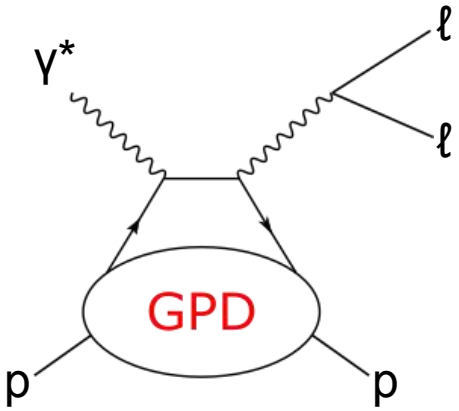
GPDs (Generalized Parton Distributions)

- 3D functions describing partonic structure of nucleon
- Each one defined for specific parton and specific helicity configuration
- Studied in various experimental channels
- pQCD involved

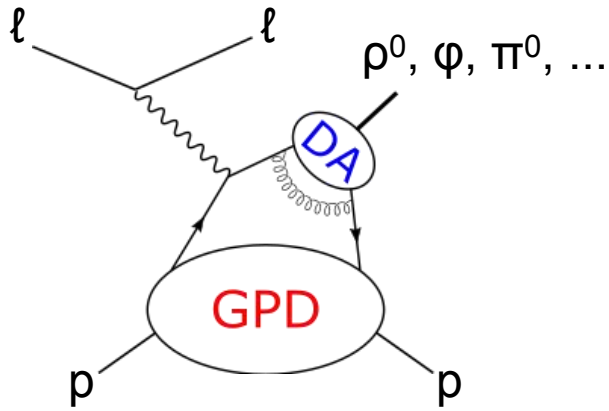
DVCS



TCS



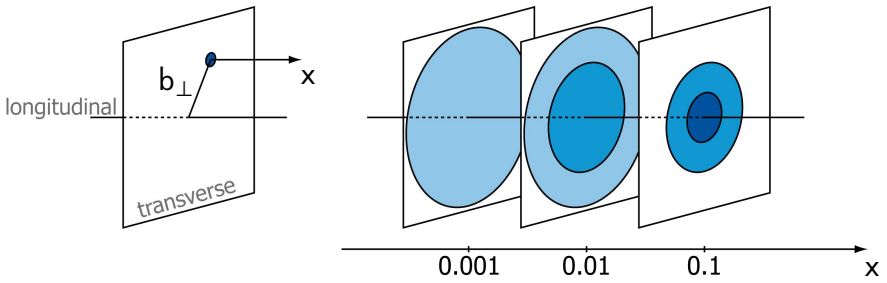
HEMP



GPDs (Generalized Parton Distributions)

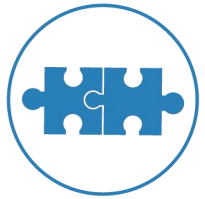
- Nucleon tomography

$$q(x, \mathbf{b}_\perp^2) = \int \frac{d^2\boldsymbol{\Delta}}{4\pi^2} e^{-i\mathbf{b}_\perp \cdot \boldsymbol{\Delta}} H^q(x, 0, t = -\boldsymbol{\Delta}^2)$$

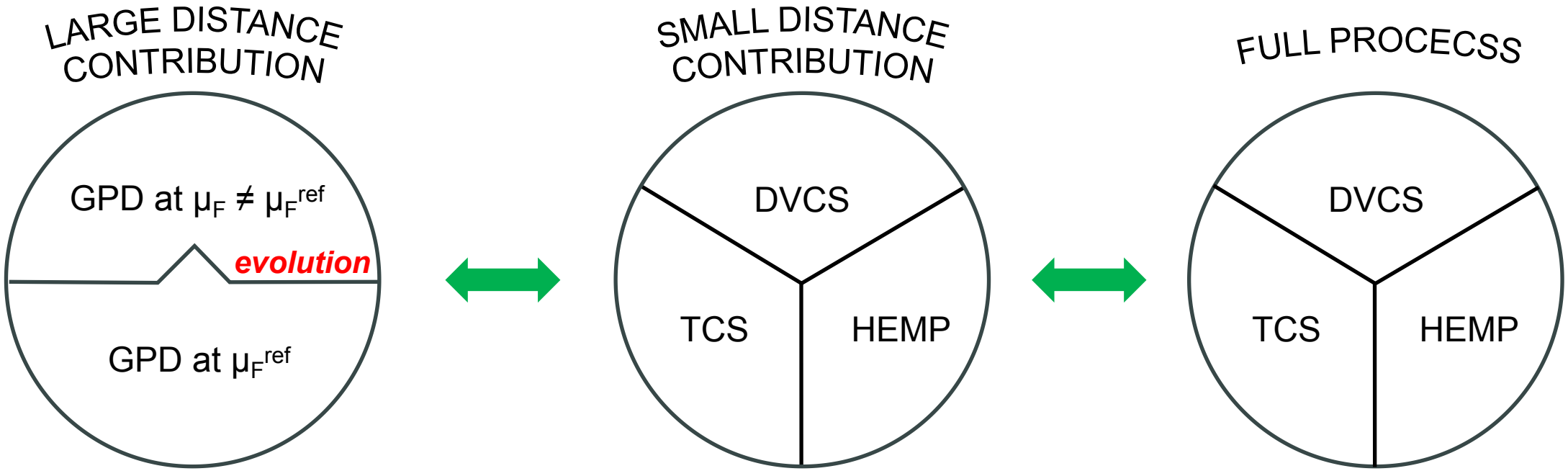


- Total angular momentum

$$\int_{-1}^{1} dx\ x\,[H^q(x, \xi, 0) + E^q(x, \xi, 0)] = 2J_q$$



S_p = 1/2



First principles and fundamental properties

Computation of amplitudes

Experimental data and phenomenology

Tasks and challenges:

- Physical models
- Perturbative approximations
- Many observables
- Numerical methods
- Accuracy and speed
- Fits

PARTONS (PARtonic Tomography Of Nucleon Software)

B. Berthou (Irfu), D. Binosi(ECT*), N. Chouika (Irfu), L. Colaneri (IPNO/U.Conn.), M. Guidal (IPNO), K. Joo (U. Conn), P. Lafitte (ECP), C. Mezrag (Argonne), H. Moutarde (Irfu), F. Sabatie (Irfu), P. Sznajder (IPNO), P. Rodríguez Quintero (UHU), J. Wagner (NCBJ Warsaw)

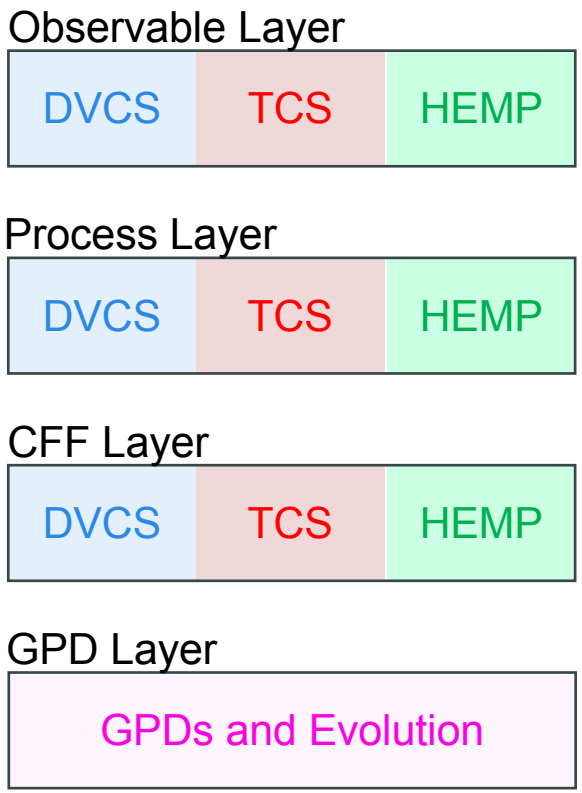
Layered structure:

- one layer = collection of objects designed for common purpose
- one module = one physical development
- operations on modules provided by Services, e.g. for GPD Layer

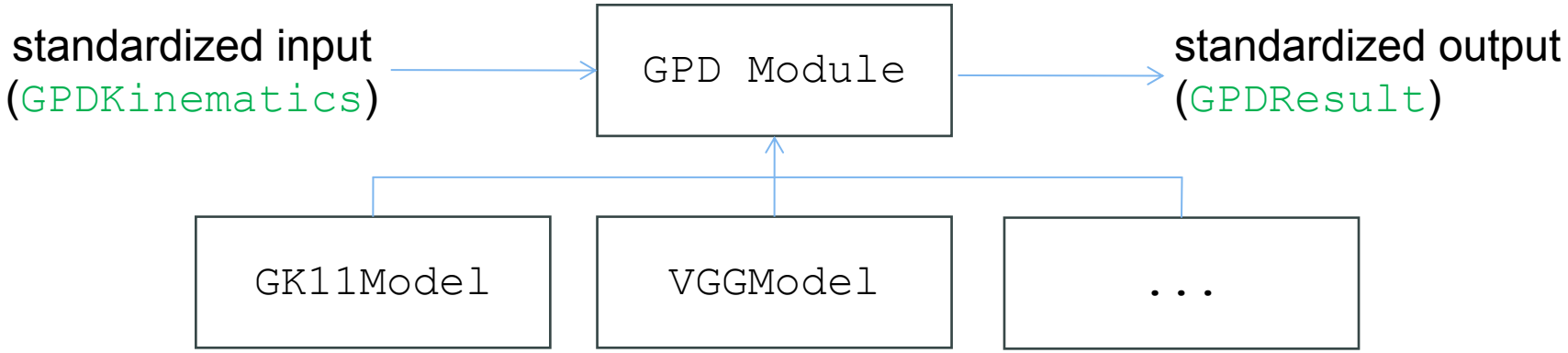
```
GPDResult computeGPDModel
        (const GPDKinematic& gpdKinematic, GPDModule* pGPDModule) const;
GPDResult computeGPDModelRestrictedByGPDType
        (const GPDKinematic& gpdKinematic, GPDModule* pGPDModule,
        GPDType::Type gpdType) const;
GPDResult computeGPDModelWithEvolution
        (const GPDKinematic& gpdKinematic, GPDModule* pGPDModule,
        GPDEvolutionModule* pEvolQCDModule) const;
...
```

- what can be automated is automated
- features improving calculation speed

  e.g. CFF Layer Service stores the last calculated values



Standardized objects:



- benefiting from C++ inheritance and polymorphism mechanisms
- reduction of mistake probability
- adding new modules as easy as possible

C++ interface or ...

```
// Retrieve GPD service
GPDService* pGPDService =
        Partons::getInstance()->getServiceObjectRegistry()->getGPDService();

// Load GPD module with BaseModuleFactory
GPDModule* pGPDModel =
        Partons::getInstance()->getModuleObjectFactory()->newGPDModule(
                GK11Model::classId);

// Create GPDKinematic(x, xi, t, MuF2, MuR2) to compute
GPDKinematic gpdKinematic(1.E-1, 1.E-2, -0.4, 2., 2.);

// Perform the calculation
GPDResult gpdResult = pGPDService->computeGPDModelRestrictedByGPDType(
        gpdKinematic, pGPDModel, GPDType::H);

// Print result
std::cout << gpdResult.toString() << std::endl;
```

```
> GPD_H
GluonDistribution = 0.47903750855896859
u = 2.75476       u(+) = 3.13425    u(-) = 2.37528
d = 1.66197       d(+) = 2.04145    d(-) = 1.28248
s = 0.183929      s(+) = 0.367857   s(-) = 0
```

XML interface (the best way to manage your computations)

```
<?xml version="1.0" encoding="UTF-8" standalone="yes" ?>

<scenario date="2016-03-25" description="Example: computation of one GPD model without evolution">

    <task service="GPDService" method="computeGPDModel" storeInDB="0">

        <kinematics type="GPDKinematic">
            <param name="x" value="1.E-1" />
            <param name="xi" value="1.E-2" />
            <param name="t" value="-0.4" />
            <param name="MuF2" value="2." />
            <param name="MuR2" value="2." />
        </kinematics>

        <computation_configuration>
            <module type="GPDModule">
                <param name="className" value="GK11Model" />
            </module>
        </computation_configuration>

    </task>

</scenario>
```

Database

- Result computed by each layer can be stored/retrieved from database

```
// Retrieve GPD DAO service
GPDResultDaoService gpdResultDaoService;

// Insert
int computationId = gpdResultDaoService.insert(gpdResult);

// Retrieve
List<GPDResult> gpdList = resultService.getGPDResultListByComputationId(computationId);
```

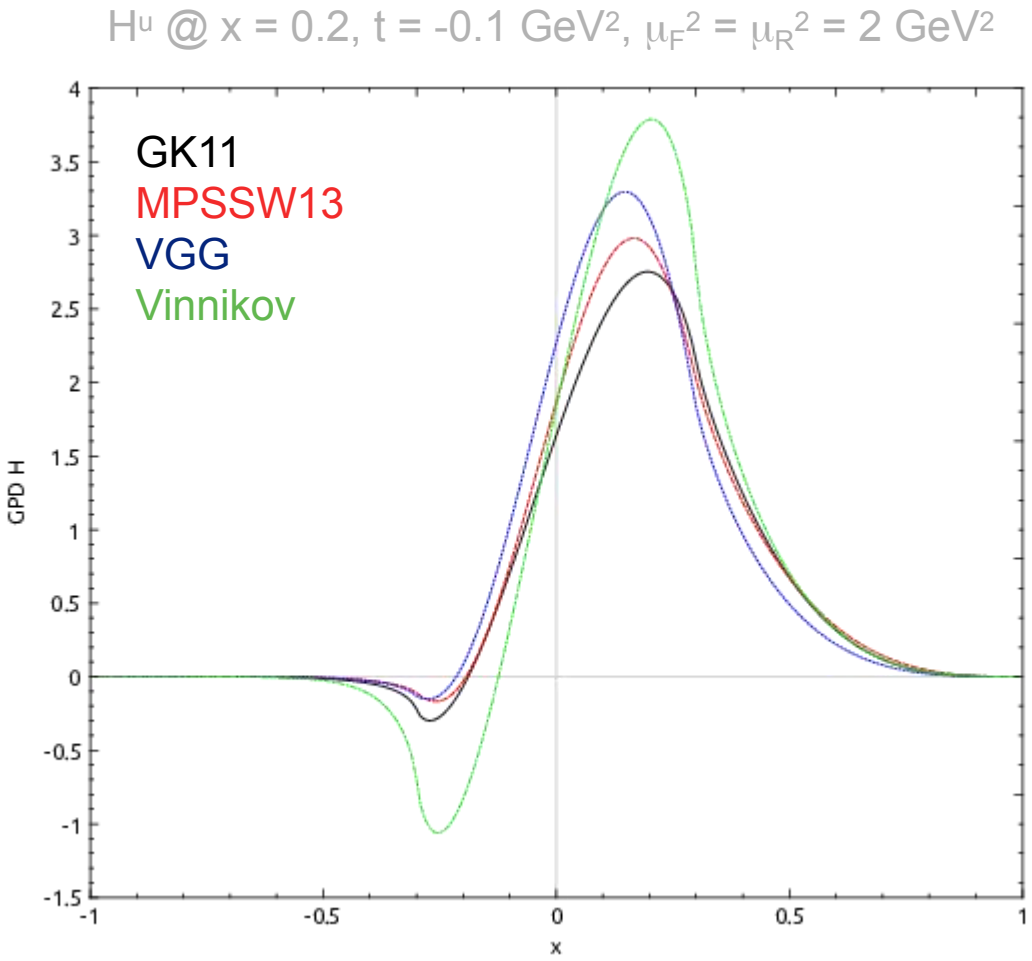
- MySQL and SQLite support
- Optimized for large transactions
- Database also to store experimental data → Fits

Threads

- To speed up calculation
- Used for instance by Logger

Existing modules:

- GPD: GK11, VGG, Vinnikov, MPSSW13, MMS13
- Evolution: Vinnikov code
- CFF (DVCS only): LO, NLO (gluons and light or light + heavy quarks)
- Cross Section (DVCS only): VGG, BMJ, GV
- Running coupling: 4-loop PDG expression, constant value



Last but not least:

- Good development practice
- Robustness and accuracy
- Non-regression tools

SUMMARY

PARTONS (PARtonic Tomography Of Nucleon Software)

- Modern platform devoted to study GPDs
- Design to support the effort of GPD community
- Can be used by both theoreticians and experimentalists

- First release expected in this year
- We kindly ask for any feedback

- More info in: arXiv: hep-ph/1512.06174