Eur. Phys. J. C manuscript No.
(will be inserted by the editor)

Open database for GPD analyses

V.D. Burkert[1], A. Camsonne[1], P. Chatagnon[1,2], K. Cichy[3], M. Constantinou[4], H. Dutrieux[5], I. M. Higuera-Angulo[1], C. Mezrag[2], D. Richards[1] P. Sznajder[6]

[1] Thomas Jefferson National Accelerator Facility, 12000 Jefferson Avenue, Newport News, VA 23606, USA
[2] Irfu, CEA, Université Paris-Saclay, F-91191 Gif-sur-Yvette, France
[3] Faculty of Physics and Astronomy, Adam Mickiewicz University, ul. Uniwersytetu Poznańskiego 2, 61-614 Poznań, Poland
[4] Department of Physics, Temple University, Philadelphia, PA 19122 - 1801, USA
[5] Physics Department, William & Mary, Williamsburg, VA 23187, USA
[6] National Centre for Nuclear Research, NCBJ, 02-093 Warsaw, Poland

March 25, 2025

Abstract This article summarizes the main ideas behind creating an open database proposed for use in the exploration of generalized parton distributions (GPDs). This lightweight database is well suited for GPD phenomenology and is designed to store both experimental and lattice-QCD data. It can also aid in benchmarking GPD-related developments, such as GPD models. The database utilizes a new data format based on the YAML serialization language, enabling the storage of essential information for modern analyses, such as replica values. It includes interfaces for both Python and C++, allowing straightforward integration with analysis codes.

1 Introduction

Generalized parton distributions (GPDs) [1–3] arise in the factorisation theorems of perturbative quantum chromodynamics. These universal objects describe the non-perturbative parts of exclusive reactions, such as deeply virtual Compton scattering (DVCS) [2, 4], and are associated with the partonic structure of hadrons. To some extent, GPDs can be considered a unification of parton distribution functions (PDFs) and elastic form factors, which are typically studied without any connection. The importance of GPDs in particle physics arises from their relation to nucleon tomography [5, 6], allowing the study of the position of partons carrying a specific fraction of a hadron's momentum, as well as their connection to the energy-momentum tensor [7, 8]. This latter feature enables the determination of the total angular momentum carried by specific partons and provides insight into the so-called "mechanical properties" of media composed of confined partons, see e.g. [9–12]. The phenomenology of GPDs, compared to that of PDFs, comes with additional difficulties. The most striking challenge arises from the three-dimensional nature of GPDs, meaning that more data are required to constrain these objects than in the case of one-dimensional PDFs. The situation is worsened by the variety of GPD types that simultaneously contribute to exclusive reactions, defined for combinations of hadron-parton spins – some of which do not even have counterparts in inclusive physics. Exclusive reactions also have relatively small cross-sections, and the requirement of reconstructing all particle states imposes non-trivial constraints on the experimental apparatus.

Despite these difficulties, a collection of data has already been measured by experiments at JLAB, DESY, and CERN (for a review, see for instance Ref. [13, 14]). These data are not only for DVCS but also for processes like deeply virtual meson production (DVMP) [15, 16], timelike Compton scattering (TCS) [17, 18], and exclusive heavy meson production [19, 20]. Several other exclusive processes have been proposed in recent years, including double deeply virtual Compton scattering (DDVCS) [21–23] and $2 \to 3$ processes [24–28]. All of the aforementioned processes, measured under various experimental conditions (such as beam and target polarization states and different particles), provide complementary information about GPDs and help to pinpoint their specific contributions. It is also important to mention that the GPD topic will be a pillar of the physics program of current and future QCD laboratories, particularly JLab [29], EIC [30], and EIcC [31].

An additional source of GPD information is provided by lattice QCD computations, traditionally via moments

of these objects extracted from local matrix elements (see, e.g., Refs. [32–38]) or, more recently, through the novel techniques utilizing non-local matrix elements, the quasi- [39] and pseudo-distribution methods [40] or the off-forward Compton amplitude approach [41]. These recent techniques have already witnessed an extensive amount of work aimed at several types of the nucleon's GPDs, see e.g. [41–51]. It is expected that lattice QCD results will be crucial in constraining GPDs in the phase space that are inaccessible (or practically difficult to access) through currently measured exclusive processes or those to be measured in the foreseeable future. The limited access to GPDs through some exclusive processes is discussed in the literature under the topic of "shadow GPDs" [52–56].

Despite the wealth of existing data sensitive to GPDs – a collection that we expect to grow significantly in the future due to upcoming experiments and lattice QCD calculations – the community still lacks an open database that could accelerate research progress. Such a centralized platform providing easy access to GPD-related data could serve several purposes. The most straightforward benefit would be saving the time needed to incorporate existing and new data into phenomenological analyses. Several groups employing various modeling strategies are currently working on global analyses of GPDs (see, for instance, Refs. [57–60]). Each of these groups could benefit from using the same database, directly fed by experimental and lattice QCD collaborations. Another advantage of using a common database would be improved reproducibility of phenomenological analyses, enabling the community to fully adopt open-science standards. Additionally, the proposed database could also be populated with pseudo-data, not only those used in impact studies but also those serving as benchmarks for theoretical developments, such as values obtained from new GPD models. This would allow others to verify that their implementation of these models is correct.

We stress that the proposed database has been designed based on the experience of GPD phenomenologists and lattice-QCD practitioners and, therefore, fully addresses the needs of future analyses aimed at exploring parton distributions. It also provides direct integration with existing codes and, in general, is much better tailored for GPD physics than other open databases used in particle physics, in particular HEPData [61], which is designed for the general purpose of storing data in high-energy physics.

In this article, we summarise the main ideas for the open database we propose for use in the GPD community. We begin with Sect. 2, which describes the essential whys and hows behind the project, including the server selection, format, programming languages, etc. In Sect. 3, we detail the data format we have adopted for this purpose. This format allows for storing both experimental and lattice QCD data and includes many useful features, such as the ability to store bin boundaries and replica values. Finally, in Sect. 4, we discuss the database interface that users can integrate into their codes, without providing too many technical details, which can be accessed elsewhere, specifically on the project's website [62]. A concise summary is provided in Sect. 5.

2 Basics

The project aims to create and maintain a database capable of storing a collection of experimental and lattice-QCD data. A specific format is used to store such data, and a dedicated library is provided in both C++ and Python, allowing users to easily access the database in their analysis codes. Additionally, a webpage [62] has been created, featuring up-to-date documentation and a list of available data.

To achieve this ambitious goal despite limited computing and human-power resources, we utilized GitHub [63], one of the basic tools used in the particle physics community. This choice offers several advantages. By using an existing service, we eliminate the need to maintain a separate database server or host a webpage. GitHub provides free hosting, with the option to mirror the project, e.g., on local GitLab instances. This helps prevent situations where access to the database is lost, such as when a university discontinues its hosting service or an administrator leaves the field. GitHub also allows the formation of a management team and offers a straightforward interface for interacting with users. These features can be used to report issues, suggest improvements, and add new data. Additionally, version control is simple and transparent. Altogether, these features enhance the chances that the database will remain useful to the community, preventing it from fading into obscurity.

The database stores files in a text format (see Sect. 3 for details). This approach simplifies database maintenance, as each data series is stored in a separate file. Adding new data is equally simple; for example, an experimental collaboration needs only to provide a new text file in the specified format to incorporate it into the database. However, the use of text data files has a drawback arising from their size. Two factors must be considered in this context: the limitations of GitHub's

service and the potential burden on users' computing systems. The initial size of all files in the database (which includes most of the world DVCS data) is only a few megabytes. We estimate that the adopted solution should be feasible for at least several years. We do not plan to store "intermediate" data produced in lattice-QCD computations (which can reach gigabytes in size), opting instead to store only "final" data that can be used in phenomenological analyses. If using GitHub becomes impractical, we can migrate the project to a different server, use a premium account, or adopt an alternative solution, such as transferring the data to a MySQL server – an option we considered when developing the data format.

The database is initially populated with the published experimental data (respecting the appropriate licenses) and some lattice-QCD results. The project's webpage is created using Jekyll technology and is automatically updated whenever the project is modified. For example, adding a new data file triggers the regeneration of the webpage, including an updated list of stored data. The Python and C++ interfaces are complete (see Sect. 4 for more details). The Python package is automatically deployed to the PyPi service [64], while the C++ interface is a wrapper for the Python code. This allows us to write essentially one codebase for use in both programming languages. Finally, a series of automated checks have been defined on GitHub, helping to reduce the risk of releasing a broken version of the project.

3 Data format

In this section, we describe the structure of the data files we intend to store in the database. For the sake of simplicity and portability, we have chosen the YAML data serialization format [65], which can be easily processed in both Python and C++ codes. YAML is an ASCII-based format, where the structure of data is defined by indentation. An example of a dummy data file presenting all available features users can use when storing data in the database is shown below. Thanks to the use of human-readable keys and additional comments (starting with '#'), the presented structure of data should be intuitive even for non-expert users. An additional description is provided below the example.

```yaml
---
uuid: 9j7gof4d # id

general_info:

   date: 2023-09-25 # date of insertion to database
   data_type: DVCS # data type (enum)
   pseudodata: true # tag to distinguish between data and pseudo-data (e.g. Monte Carlo study)
        # (optional) if not set pseudodata = false
   collaboration: "Example" # collaboration name releasing data stored (limited to 40 characters)
   reference: "arXiv:01/02" # reference, (limited to 255 characters)
   conditions: # experimental or lattice-QCD conditions
      lepton_beam_type: e- # lepton beam type (enum)
      lepton_beam_energy: 10. # lepton beam energy (by default in GeV)
      hadron_beam_type: p # hadron beam type (enum)
      hadron_beam_energy: 100. # hadron beam energy (by default in GeV)
        # if missing, fixed target assumed
   comment: "Comment" # comment (optional, limited to 255 characters)

data:

    - data_set:

        label: "Q2_dep" # label of current data set
                        # limited to 40 characters

        kinematics: # data related to kinematics

            name: [xB, Q2] # definition of kinematic phase-space,
                           # here 2D (xB, Q2) domain (enums)
            unit: [none, GeV2] # units (enums)
            value: # mean values of xB and Q2 obtained in three kinematic bins
                - [0.2, 1.5] # 1st bin: xB = 0.2, Q2/GeV2 = 1.5
                - [0.3, 2.2] # ...
                - [0.4, 3.1] # 3th bin: xB = 0.4, Q2/GeV2 = 3.1
            # or alternatively:
            # value: [[0.2, 1], [0.3, 2], [0.4, 3]]

            unc: # uncertainties associated to the mean values (optional)
                - [0.0015, 0.016] # 1st bin: xB = 0.2 +- 0.0015, Q2/GeV2 = 1.5 +- 0.016
                - [0.0012, 0.011] # ...
                - [0.0019, 0.013] # 3th bin: xB = 0.4 +- 0.0019, Q2/GeV2 = 3.1 +- 0.013
            bin: # bin boundaries (optional)
                - [[0.11, 0.25], [1, 2]] # 1st bin: 0.11 < xB < 0.25 and 1 < Q2/GeV2 < 2
                - [[0.25, 0.35], [2, 3]] # ...
                - [[0.35, 0.52], [3, 4]] # 1st bin: 0.35 < xB < 0.52 and 3 < Q2/GeV2 < 4
            replica: # values obtained from replicas used in the related analysis (optional)
                     # in the first bin three replicas give:
```

```
                # xB = {0.2005, 0.1967, 0.1995}
                ↪ and Q2 = {1.514, 1.502,
                ↪ 1.487}
                # the same replicas in the
                ↪ second bin give:
                # xB = {0.2997, 0.3005, 0.2988}
                ↪ and Q2 = {2.212, 2.193,
                ↪ 2.207}
            - [[0.2005, 0.1967, 0.1995], [1.514,
            ↪ 1.502, 1.487]]
            - [[0.2997, 0.3005, 0.2988], [2.212,
            ↪ 2.193, 2.207]]
            - [[0.4002, 0.4003, 0.3987], [3.094,
            ↪ 3.094, 3.107]]

    observable: # data related to observables

        name: [ALU, ALL] # definition of
        ↪ observables (enums)
                        # here, in each
                        ↪ kinematic bin
                        ↪ experiment measures
                        ↪ two observables:
                        # ALU and ALL
                        ↪ asymmetries
        unit: [none, none] # units (enums)
        value: # mean values of ALU and ALL
        ↪ obtained in three kinematic bins
            - [0.13, 0.29] # 1st bin: ALU = 0.13,
            ↪ ALL = 0.29
            - [0.14, 0.24] # ...
            - [0.11, 0.26] # 3th bin: ALU = 0.11,
            ↪ ALL = 0.26
        stat_unc: # statistical uncertainties
        ↪ associated to the mean values
        ↪ (optional)
            - [0.03, 0.08] # 1st bin: ALU = 0.13
            ↪ +- 0.03, ALL = 0.29 +- 0.08
            - [0.04, 0.07] # ...
            - [0.03, 0.02] # 3th bin: ALU = 0.11
            ↪ +- 0.03, ALL = 0.26 +- 0.02
        sys_unc: # systematic uncertainties
        ↪ associated to the mean values
        ↪ (optional)
                # here we demonstrate how one
                ↪ can use asymmetric
                ↪ uncertainties
                # and can specify a correlation
                ↪ between uncertainties
            - [0.02, 0.01] # 1st bin: ALU = 0.13
            ↪ +- 0.02, ALL = 0.29 +- 0.01
            - [0.01, [0.02, 0.03]] # 2nd bin: ALU
            ↪ = 0.14 +- 0.01, ALL = 0.24 - 0.02
            ↪ + 0.03
            - ["corr_matrix1", 0.01, 0.02]
                # 3th bin: ALU = 0.11 +- 0.01,
                ↪ ALL = 0.26 +- 0.02,
                # uncertainties are correlated
                ↪ according to
                # 'corr_matrix1' matrix, see
                ↪ 'correlation' section
        sys_unc_contrib_label: ["fit",
        ↪ "detector"]
                # labels of contributions to
                ↪ systematic
                # uncertainties (optional)
                # limited to 40 characters
        sys_unc_contrib: # contributions to
        ↪ systematic uncertainties (optional)
            - [[0.015, 0.013], [0.007, 0.007]]
                # contributions to systematic
                ↪ uncertainties of
                # asymmetries measured in the
                ↪ first bin,
                # i.e. 0.02 and 0.01 values (see
                ↪ above)
                # contributions are:
                # 0.015 and 0.007 for source
                ↪ labeled as 'fit'
                # 0.013 and 0.007 for source
                ↪ labeled as 'detector'
            - [[0.007, 0.008], [[0.010, 0.017],
            ↪ [0.020, 0.023]]] # use of
            ↪ asymmetric uncertainties
            - ["corr_matrix2", 0.005, 0.008,
            ↪ 0.014, 0.016]
                # use of correlation matrix with
                ↪ elements:
                # sigma_sys_ALU_fit
                # sigma_sys_ALU_detector
                # sigma_sys_ALL_fit
                # sigma_sys_ALL_detector
        replica: # values obtained from replicas
        ↪ used in the related analysis
        ↪ (optional)
                # in the first bin three
                ↪ replicas give:
                # ALU = {0.132, 0.127, 0.140}
                ↪ and ALL = {0.295, 0.237,
                ↪ 0.327}
                # the same replicas in the
                ↪ second bin give
                # ALU = {0.137, 0.149, 0.105}
                ↪ and ALL = {0.183, 0.232,
                ↪ 0.306}
            - [[0.132, 0.127, 0.140], [0.295,
            ↪ 0.237, 0.327]]
            - [[0.137, 0.149, 0.105], [0.183,
            ↪ 0.232, 0.306]]
            - [[0.097, 0.113, 0.120], [0.259,
            ↪ 0.262, 0.259]]
        norm_unc: [0.001, 0.002]
                # normalisation uncertainties
                ↪ (optional)
                # ALU +- 0.001, ALL +- 0.002 (for
                ↪ each kinematic bin)
        norm_unc_contrib_label: ["target_pol",
        ↪ "beam_pol"]
                # labels of contributions to
                # systematic uncertainties
                ↪ (optional)
                # limited to 40 characters
        norm_unc_contrib: [[0, 0.001], [0.0014,
        ↪ 0.0019]]
                # contributions to systematic
                # uncertainties (optional)
                # contributions are:
                # 0.0006 and 0.0014 for source
                ↪ labeled as 'target_pol'
                # 0.0008 and 0.0019 for source
                ↪ labeled as 'beam_pol'
```

```
correlation: #definition of correlation matrices used
↪  in data file

    - ["corr_matrix1", 1, 0.2, 0.2, 1]
                    # 2D matrix labeled as
                    ↪  'corr_matrix1',
                    # with row-by-row values
                    ↪  ((1,0.2),(0.2,1))
    - ["corr_matrix2", 1, 0.1, 0.3, 0.2, 0.1, 1,
    ↪  0.5, 0.1, 0.3, 0.5, 1, -0.2, 0.2, 0.1, -0.2,
    ↪  1] # 4D matrix
```

All data files are made out of three to four sections (`correlation` section is optional), see Fig. 1. These are:

- `uuid`: data file identifier
- `general_info`: general information about data stored (metadata)
- `data`: data stored
- `correlation`: definition of correlation matrices used throughout `data` section

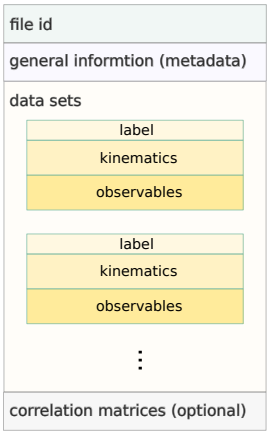

Fig. 1 General structure of data files.

A single data file can store multiple data sets via `data_set` subsections, each one containing:

- `label`: data set label
- `kinematics`: kinematics stored
- `observable`: observables stored

The idea behind storing multiple data sets in a single data file is that experiments often use the same data to separately extract observables as functions of independent variables. For instance, in Ref. [66], COMPASS reports the measurement of $A_{\rm UT}$ asymmetry for deeply virtual meson production in 1D bins of x_B, Q^2 or $p_T^2 \approx -t$, based on raw data obtained in the same experimental runs. Therefore, in this case, it makes sense to store three different data sets in a single data file. These data sets must be distinguished by unique labels, such as `"xB_dep"`, `"Q2_dep"` and `"pT2_dep"`, so that users may unambiguously choose which data sets (and, therefore, kinematic dependencies) they want to use in their analyses. In the case of lattice-QCD results, many observables are computed on the same set of samples of the QCD vacuum – the so-called gauge configurations – and therefore exhibit correlations among themselves. It is useful to group such observables together to allow a proper handling of those uncertainties through replicas.

The presented data file defines just one data set labeled `"Q2_dep"`. The stored data are the $A_{\rm LU}$ and $A_{\rm LL}$ asymmetries measured in 2D bins of x_B and Q^2. Three kinematic points are defined, with the mean values of kinematic variables and measured observables, together with uncertainties, specified in Table 1. In our example, all statistical uncertainties are symmetric. On the other hand, systematic uncertainties are symmetric only for the first point. For the second point, one of the uncertainties is asymmetric. In contrast, for the third point, the uncertainties are correlated according to `corr_1` correlation matrix defined in `correlation` section of the data file. In addition to the statistical and systematic uncertainties, we also have information on normalization uncertainties, which are typically related to the polarization measurements. For a given observable, this type of uncertainty is common to all data points. We note that keeping a detailed track of various types of uncertainties is important for further propagation, as each type of uncertainty is typically handled differently in phenomenological analyses; see, for instance, details of the replication method described in Ref. [67].

In addition to the basics described so far, we have additional information stored in our exemplary data file. For kinematics, the definition of bins used to extract observables is provided. For instance, observables measured at the first kinematic point were extracted in the following 2D bin: $0.11 < x_B < 0.25$ and $1\ \text{GeV}^2 < Q^2 < 2\ \text{GeV}^2$. Such information is useful not only for bookkeeping but also allows for more precise comparisons between experimental data and theory. Moreover, in some cases, like in Ref. [68], experiments only provide the definition of bins and the cross-section integrated within these bins.

In e.g. lattice-QCD computations, the mean values of kinematic variables and related uncertainties are usually estimated from a set of replicas. Values given by these replicas can be stored in the data files via `replica` key. In our example, the first replica in the first bin gives $\{x_B = 0.2005, Q^2 = 1.514\ \text{GeV}^2\}$, the second

Table 1 Three data points stored in the exemplary data file. In addition to the kinematics (mean values of x_B and Q^2) and the values of the measured observables, statistical (σ_{stat}), systematic (σ_{sys}) and normalization (σ_{norm}) uncertainties are also provided. The quantities marked with an asterisk are correlated, with a correlation factor of 0.2.

x_B	Q^2 [GeV^2]	A_{LU}				A_{LL}			
		value	σ_{stat}	σ_{sys}	σ_{norm}	value	σ_{stat}	σ_{sys}	σ_{norm}
0.2	1.5	0.13	± 0.03	± 0.02	± 0.001	0.29	± 0.08	± 0.01	± 0.002
0.3	2.2	0.14	± 0.04	± 0.01	"	0.24	± 0.07	$^{+0.03}_{-0.02}$	"
0.4	3.1	0.11	± 0.03	± 0.01*	"	0.26	± 0.02	± 0.02*	"

$\{x_B = 0.1967, Q^2 = 1.502\ \text{GeV}^2\}$, etc. For observables, in addition to the information provided by individual replicas, we may store information on various contributions to the uncertainties. These contributions are distinguished by labels. For instance, as shown in Table 1, the normalization uncertainty associated with all A_{LL} points is '0.002'. With additional information provided by `norm_unc_contrib` key, we know that the contribution of beam polarization uncertainty to this value is `0.0014`, while the contribution of target polarisation uncertainty is `0.0019`. For A_{LU}, only the beam polarisation uncertainty contributes.

The values of keys such as `data_type`, `unit`, and `hadron_beam_type` are predefined, resembling the concept of enumerators known in C++. This approach helps maintain database coherence and prevents pollution from inconsistent nomenclature. For instance, the library will process the value `unit: xB` without interruption but will trigger an error for `unit: xBj`. This feature is also crucial for users, as they can build their code around the library without needing to create a separate dictionary for each data file. The enumerators are defined in separate YAML files, making it easy to add new values. For example, predefined data types, which users must specify using the `data_type` key, are defined in the following file:

```yaml
---
data:

  # structure function (like elastic FFs, F2, etc.)
  - name: STRUCTURE_FUNCTION
    description: "structure function"
    required_name: [ hadron_type ]
    required_type: [ particle ]
  # elastic
  - name: LATTICE_QCD
    description: "lattice QCD"
    required_name: [ hadron_type, pion_mass,
    ↪  lattice_spacing ]
    required_type: [ particle, float, float ]
  # exclusive: DVCS
  - name: DVCS
    description: "deeply virtual Compton scattering"
    required_name: [ lepton_beam_type,
    ↪  lepton_beam_energy, hadron_beam_type,
    ↪  hadron_beam_energy ]
    required_type: [ particle, float, particle,
    ↪  float ]
```

This example also demonstrates another feature of the database library, namely, type checks. For instance, `data_type:DVCS` requires defining `lepton_beam_type`, `lepton_beam_energy` and `hadron_beam_type` in the `conditions` section. However, not just any data can be specified there. For example, `lepton_beam_energy` must be a float value, while `lepton_beam_type` must be a particle type, such as `p` or `H4`, that is recognized by the "particle" library [69]. Providing an incorrect data type will result in an error triggered during the processing of such a corrupted data file. Several additional checks are implemented in the library. For instance, when defining an observable, one must specify the associated unit type. The predefined unit types include, for instance, `EVm2`, which applies to all units that can be converted to eV^{-2} in the natural system, such as barns and GeV^{-2}, and `ANGLE`, applicable to radians, degrees, etc. This solution prevents setting, for example, $Q^2 = 2$ GeV, and also opens the possibility of implementing a unit conversion at a later stage of the project. It is worth noting that the use of the "particle" library allows easy access to information such as mass and charges of particles, which can be straightforwardly used in users' analysis codes.

Alphanumeric data, such as those associated with the `collaboration` or `label` keys, are limited to a certain number of characters (see comments in the example data file). This solution addresses practical concerns of storage, performance and data integrity, and is routinely implemented in SQL-like databases. Since we foresee using an SQL format in the future as an alternative to YAML files stored on GitHub, it seems appropriate to comply with SQL practices now. Aside from the length check, alphanumeric data are not validated in any other way. It is, therefore, up to the users and admins to insert meaningful data of this type. Additional information to help understand the content of the data file can be inserted as com-

ments, i.e., via the `comment` key or the '#' YAML tag. This may be particularly helpful in deciphering labels set via the `label`, `sys_unc_contrib_label`, and `norm_unc_contrib_label` keys. For the `uuid` key, serving as an identifier of the data file, we intend to use short universally unique identifiers, which can be easily generated in Python or even dedicated web-pages, like [70].

We stress that, if necessary, the data format can be modified in the future to include additional information while maintaining full backward compatibility. In particular, for lattice QCD, we highlight the possibility of adding indicators of data quality. Criteria for this purpose have been developed over many years by the Flavour Lattice Averaging Group (FLAG) [71], and could be adopted for the case of structure functions, taking into account specific challenges unique to pseudo- and quasi-distribution approaches, such as solving the inverse moment problem. The introduction of such criteria and a transparent method for assigning them would facilitate comparisons between calculations performed under various conditions and approaches, enabling a meaningful global analysis of lattice-QCD results.

4 Users interface

A library is provided to check the database content, load a given data file, and access stored information. The library is written in Python, allowing for its straightforward use in analysis codes. A C++ wrapper for this library, based on the generic Python.h library, is also available. This enables the database to be used in projects like GeParD [72] and PARTONS [73], which are written in Python and C++, respectively. The Python library is available in the PyPI repository, making its installation particularly easy – typically, the installation is reduced to executing only one command:

```
pip3 install gpddatabase
```

The code of C++ wrapper is available in the main repository of the project [74], and it requires the Python library to be pre-installed. The wrapper can be easily incorporated in any C++ projects thanks to the provided CMake [75] module.

The following example demonstrates how to access the database using the Python library. For simplicity, we present only the most basic operations for the Python interface. Examples, including those for C++, and technical documentation are available on the project's main page.

```python
# import module
import gpddatabase

# make a reference to the database
db = gpddatabase.ExclusiveDatabase()

# print availible uuids
print(db.get_uuids())

# load a given data file
ob = db.get_data_object('9j7gof4d')

# print date of insertion stored in general_info
↪  section
print(ob.get_general_info().get_date())

# print labels of availible datasets
print(ob.get_data().get_data_set_labels())

# print number of points stored in 'Q2_dep' dataset
print(ob.get_data().get_data_set('Q2_dep').
   get_number_of_data_points())

# make a referenece to the first point
point = ob.get_data().get_data_set('Q2_dep').
   get_data_point(0)

# print names of kinematic variables, then units and
↪  values
print(point.get_kinematics_names())
print(point.get_kinematics_units())
print(point.get_kinematics_values())

# print names of observables, then units and values
print(point.get_observables_names())
print(point.get_observables_units())
print(point.get_observables_values())
```

A local installation of the library involves downloading all data files available in the database. The location of these files can be checked in the following way.

```python
# import module
import gpddatabase

# make a reference to the database
db = gpddatabase.ExclusiveDatabase()

# print location of data files
print(db.get_path_to_databse())
```

This is a straightforward solution that simplifies the use of the database (no need to manually select and download the files) and allows, for instance, easy addition and testing of new content. As already mentioned in Sect. 2, the current size of the library is only a few megabytes but will grow in the future. Therefore, switching to more elaborate ways of managing data files, such as those implemented in the LHAPDF library [76], is an option to be considered for future releases of the project.

5 Summary

This article summarizes the key concepts behind the open database we propose for use in the GPD community. This lightweight database is well suited for GPD phenomenology and is designed to store both experimental and lattice-QCD data. It can also be used to benchmark GPD-related developments, such as GPD models. The database utilizes a new data format based on the YAML serialization language, which enables storage of essential information for modern phenomenology analyses, including replica values, bin sizes, correlation matrices, and more. Predefined types are used to ensure data consistency. A user interface is provided, allowing straightforward integration with analysis codes in Python and C++. Additional technical details on the project are available on the project's main page [62].

Acknowledgements The authors are grateful to K. Kumerički, S. Liuti, W. Melnitchouk, E. Moffat and A. Prokudin for valuable discussions. This work has been supported in part, by l'Agence Nationale de la Recherche (ANR), project ANR-23-CE31-0019. For the purpose of open access, the author has applied a CC-BY public copyright licence to any Author Accepted Manuscript (AAM) version arising from this submission. K. C. is supported by the National Science Centre (Poland) grant OPUS No. 2021/43/B/ST2/00497. M. C. acknowledges financial support by the U.S. Department of Energy, Office of Nuclear Physics under Grant No. DE-SC0025218. The work of P. S. was supported by the Grant No. 2024/53/B/ST2/00968 of the National Science Centre, Poland.

References

1. D. Müller, D. Robaschik, B. Geyer, F.M. Dittes, J. Hořejši, Fortsch. Phys. **42**, 101 (1994). DOI 10.1002/prop.2190420202
2. X.D. Ji, Phys. Rev. D **55**, 7114 (1997). DOI 10.1103/PhysRevD.55.7114
3. A.V. Radyushkin, Phys. Lett. B **380**, 417 (1996). DOI 10.1016/0370-2693(96)00528-X
4. A.V. Belitsky, D. Mueller, A. Kirchner, Nucl. Phys. B **629**, 323 (2002). DOI 10.1016/S0550-3213(02)00144-X
5. M. Burkardt, Phys. Rev. D **66**, 114005 (2002). DOI 10.1103/PhysRevD.66.114005
6. M. Burkardt, Int. J. Mod. Phys. A **18**, 173 (2003). DOI 10.1142/S0217751X03012370
7. X.D. Ji, Phys. Rev. Lett. **78**, 610 (1997). DOI 10.1103/PhysRevLett.78.610
8. M.V. Polyakov, P. Schweitzer, Int. J. Mod. Phys. A **33**(26), 1830025 (2018). DOI 10.1142/S0217751X18300259
9. V.D. Burkert, L. Elouadrhiri, F.X. Girod, Nature **557**(7705), 396 (2018). DOI 10.1038/s41586-018-0060-z
10. V.D. Burkert, L. Elouadrhiri, F.X. Girod, C. Lorcé, P. Schweitzer, P.E. Shanahan, Rev. Mod. Phys. **95**(4), 041002 (2023). DOI 10.1103/RevModPhys.95.041002
11. B. Duran, et al., Nature **615**(7954), 813 (2023). DOI 10.1038/s41586-023-05730-4
12. V.D. Burkert, L. Elouadrhiri, F.X. Girod, (2023)
13. K. Kumericki, S. Liuti, H. Moutarde, Eur. Phys. J. A **52**(6), 157 (2016). DOI 10.1140/epja/i2016-16157-3
14. N. d'Hose, S. Niccolai, A. Rostomyan, Eur. Phys. J. A **52**(6), 151 (2016). DOI 10.1140/epja/i2016-16151-9
15. J.C. Collins, L. Frankfurt, M. Strikman, Phys. Rev. D **56**, 2982 (1997). DOI 10.1103/PhysRevD.56.2982
16. K. Passek-K., Acta Phys. Polon. Supp. **16**(7), 7 (2023). DOI 10.5506/APhysPolBSupp.16.7-A5
17. E.R. Berger, M. Diehl, B. Pire, Eur. Phys. J. C **23**, 675 (2002). DOI 10.1007/s100520200917
18. P. Chatagnon, et al., Phys. Rev. Lett. **127**(26), 262501 (2021). DOI 10.1103/PhysRevLett.127.262501
19. D.Y. Ivanov, A. Schafer, L. Szymanowski, G. Krasnikov, Eur. Phys. J. C **34**(3), 297 (2004). DOI 10.1140/epjc/s2004-01712-x. [Erratum: Eur.Phys.J.C 75, 75 (2015)]
20. C.A. Flett, J.A. Gracey, S.P. Jones, T. Teubner, JHEP **08**, 150 (2021). DOI 10.1007/JHEP08(2021)150
21. A.V. Belitsky, D. Mueller, Phys. Rev. Lett. **90**, 022001 (2003). DOI 10.1103/PhysRevLett.90.022001
22. M. Guidal, M. Vanderhaeghen, Phys. Rev. Lett. **90**, 012001 (2003). DOI 10.1103/PhysRevLett.90.012001
23. K. Deja, V. Martinez-Fernandez, B. Pire, P. Sznajder, J. Wagner, Phys. Rev. D **107**(9), 094035 (2023). DOI 10.1103/PhysRevD.107.094035
24. G. Duplančić, K. Passek-Kumerički, B. Pire, L. Szymanowski, S. Wallon, JHEP **11**, 179 (2018). DOI 10.1007/JHEP11(2018)179
25. R. Boussarie, B. Pire, L. Szymanowski, S. Wallon, JHEP **02**, 054 (2017). DOI 10.1007/JHEP02(2017)054. [Erratum: JHEP 10, 029 (2018)]
26. O. Grocholski, B. Pire, P. Sznajder, L. Szymanowski, J. Wagner, Phys. Rev. D **104**(11), 114006 (2021). DOI 10.1103/PhysRevD.104.114006
27. O. Grocholski, B. Pire, P. Sznajder, L. Szymanowski, J. Wagner, Phys. Rev. D **105**(9), 094025 (2022). DOI 10.1103/PhysRevD.105.094025
28. J.W. Qiu, Z. Yu, Phys. Rev. D **109**(7), 074023 (2024). DOI 10.1103/PhysRevD.109.074023
29. A. Accardi, et al., Eur. Phys. J. A **60**(9), 173 (2024). DOI 10.1140/epja/s10050-024-01282-x
30. R. Abdul Khalek, et al., Nucl. Phys. A **1026**, 122447 (2022). DOI 10.1016/j.nuclphysa.2022.122447
31. D.P. Anderle, et al., Front. Phys. (Beijing) **16**(6), 64701 (2021). DOI 10.1007/s11467-021-1062-0
32. P. Hagler, et al., Phys. Rev. D **77**, 094502 (2008). DOI 10.1103/PhysRevD.77.094502
33. G.S. Bali, S. Collins, M. Göckeler, R. Rödl, A. Schäfer, A. Sternbeck, Phys. Rev. D **100**(1), 014507 (2019). DOI 10.1103/PhysRevD.100.014507
34. C. Alexandrou, S. Bacchio, M. Constantinou, J. Finkenrath, K. Hadjiyiannakou, K. Jansen, G. Koutsou, H. Panagopoulos, G. Spanoudes, Phys. Rev. D **101**(9), 094513 (2020). DOI 10.1103/PhysRevD.101.094513
35. E. Shintani, K.I. Ishikawa, Y. Kuramashi, S. Sasaki, T. Yamazaki, Phys. Rev. D **99**(1), 014510 (2019). DOI 10.1103/PhysRevD.99.014510. [Erratum: Phys.Rev.D 102, 019902 (2020)]
36. C. Alexandrou, et al., Phys. Rev. D **107**(5), 054504 (2023). DOI 10.1103/PhysRevD.107.054504
37. C. Alexandrou, S. Bacchio, M. Constantinou, K. Hadjiyiannakou, K. Jansen, G. Koutsou, Phys. Rev. D **104**, 074503 (2021). DOI 10.1103/PhysRevD.104.074503
38. C. Alexandrou, S. Bacchio, M. Constantinou, J. Finkenrath, R. Frezzotti, B. Kostrzewa, G. Koutsou, G. Spanoudes, C. Urbach, Phys. Rev. D **109**(3), 034503 (2024). DOI 10.1103/PhysRevD.109.034503

39. X. Ji, Phys. Rev. Lett. **110**, 262002 (2013). DOI 10.1103/PhysRevLett.110.262002
40. A.V. Radyushkin, Phys. Rev. D **96**(3), 034025 (2017). DOI 10.1103/PhysRevD.96.034025
41. A. Hannaford-Gunn, K.U. Can, J.A. Crawford, R. Horsley, P.E.L. Rakow, G. Schierholz, H. Stüben, R.D. Young, J.M. Zanotti, Phys. Rev. D **110**(1), 014509 (2024). DOI 10.1103/PhysRevD.110.014509
42. C. Alexandrou, K. Cichy, M. Constantinou, K. Hadjiyiannakou, K. Jansen, A. Scapellato, F. Steffens, Phys. Rev. Lett. **125**(26), 262001 (2020). DOI 10.1103/PhysRevLett.125.262001
43. C. Alexandrou, K. Cichy, M. Constantinou, K. Hadjiyiannakou, K. Jansen, A. Scapellato, F. Steffens, Phys. Rev. D **105**(3), 034501 (2022). DOI 10.1103/PhysRevD.105.034501
44. S. Bhattacharya, K. Cichy, M. Constantinou, J. Dodson, X. Gao, A. Metz, S. Mukherjee, A. Scapellato, F. Steffens, Y. Zhao, Phys. Rev. D **106**(11), 114512 (2022). DOI 10.1103/PhysRevD.106.114512
45. K. Cichy, et al., Acta Phys. Polon. Supp. **16**(7), 7 (2023). DOI 10.5506/APhysPolBSupp.16.7-A6
46. S. Bhattacharya, K. Cichy, M. Constantinou, X. Gao, A. Metz, J. Miller, S. Mukherjee, P. Petreczky, F. Steffens, Y. Zhao, Phys. Rev. D **108**(1), 014507 (2023). DOI 10.1103/PhysRevD.108.014507
47. S. Bhattacharya, K. Cichy, M. Constantinou, J. Dodson, A. Metz, A. Scapellato, F. Steffens, Phys. Rev. D **108**(5), 054501 (2023). DOI 10.1103/PhysRevD.108.054501
48. J. Holligan, H.W. Lin, Phys. Rev. D **110**(3), 034503 (2024). DOI 10.1103/PhysRevD.110.034503
49. S. Bhattacharya, K. Cichy, M. Constantinou, A. Metz, N. Nurminen, F. Steffens, Phys. Rev. D **110**(5), 054502 (2024). DOI 10.1103/PhysRevD.110.054502
50. H. Dutrieux, R.G. Edwards, C. Egerer, J. Karpie, C. Monahan, K. Orginos, A. Radyushkin, D. Richards, E. Romero, S. Zafeiropoulos, JHEP **08**, 162 (2024). DOI 10.1007/JHEP08(2024)162
51. S. Bhattacharya, K. Cichy, M. Constantinou, X. Gao, A. Metz, J. Miller, S. Mukherjee, P. Petreczky, F. Steffens, Y. Zhao, JHEP **01**, 146 (2025). DOI 10.1007/JHEP01(2025)146
52. V. Bertone, H. Dutrieux, C. Mezrag, H. Moutarde, P. Sznajder, Phys. Rev. D **103**(11), 114019 (2021). DOI 10.1103/PhysRevD.103.114019
53. H. Dutrieux, H. Dutrieux, O. Grocholski, O. Grocholski, H. Moutarde, H. Moutarde, P. Sznajder, P. Sznajder, Eur. Phys. J. C **82**(3), 252 (2022). DOI 10.1140/epjc/s10052-022-10211-5. [Erratum: Eur.Phys.J.C 82, 389 (2022)]
54. E. Moffat, A. Freese, I. Cloët, T. Donohoe, L. Gamberg, W. Melnitchouk, A. Metz, A. Prokudin, N. Sato, Phys. Rev. D **108**(3), 036027 (2023). DOI 10.1103/PhysRevD.108.036027
55. M.J. Riberdy, H. Dutrieux, C. Mezrag, P. Sznajder, Eur. Phys. J. C **84**(2), 201 (2024). DOI 10.1140/epjc/s10052-024-12513-2
56. K. Cichy, M. Constantinou, P. Sznajder, J. Wagner, Phys. Rev. D **110**(11), 114025 (2024). DOI 10.1103/PhysRevD.110.114025
57. K. Kumerički, D. Müller, EPJ Web Conf. **112**, 01012 (2016). DOI 10.1051/epjconf/201611201012
58. H. Moutarde, P. Sznajder, J. Wagner, Eur. Phys. J. C **78**(11), 890 (2018). DOI 10.1140/epjc/s10052-018-6359-y
59. Y. Guo, X. Ji, M.G. Santiago, K. Shiells, J. Yang, JHEP **05**, 150 (2023). DOI 10.1007/JHEP05(2023)150
60. B. Kriesten, S. Liuti, A. Meyer, Phys. Lett. B **829**, 137051 (2022). DOI 10.1016/j.physletb.2022.137051
61. HEPData development team. HEPData – repository for publication-related high-energy physics data. https://www.hepdata.net. Accessed: 28/08/2024
62. Open GPD Database development team. Open gpd database (main page). https://opengpd.github.io/gpddatabase. Accessed: 28/08/2024
63. GitHub development team. GitHub service. https://github.com. Accessed: 28/08/2024
64. PyPI development team. PyPI service – the Python package index. https://pypi.org. Accessed: 28/08/2024
65. YAML. Yaml ain't markup language (yaml™) version 1.2. https://yaml.org. Accessed: 28/08/2024
66. C. Adolph, et al., Phys. Lett. B **731**, 19 (2014). DOI 10.1016/j.physletb.2014.02.005
67. H. Moutarde, P. Sznajder, J. Wagner, Eur. Phys. J. C **79**(7), 614 (2019). DOI 10.1140/epjc/s10052-019-7117-5
68. R. Akhunzyanov, et al., Phys. Lett. B **793**, 188 (2019). DOI 10.1016/j.physletb.2019.04.038. [Erratum: Phys.Lett.B 800, 135129 (2020)]
69. Particle library development team. Particle: PDG particle data and identification codes. https://pypi.org/project/particle. Accessed: 28/08/2024
70. Online UUID Generator development team. Online UUID Generator. https://www.uuidgenerator.net. Accessed: 28/08/2024
71. Y. Aoki, et al., (2024)
72. GeParD development team. GeParD: tool for studying the 3D quark and gluon distributions in the nucleon. https://gepard.phy.hr. Accessed: 28/08/2024
73. B. Berthou, et al., Eur. Phys. J. C **78**(6), 478 (2018). DOI 10.1140/epjc/s10052-018-5948-0
74. Open GPD Database development team. Open gpd database (code repository). https://github.com/opengpd/gpddatabase. Accessed: 28/08/2024
75. CMake development team. GeParD: software for build automation. https://cmake.org. Accessed: 28/08/2024
76. A. Buckley, J. Ferrando, S. Lloyd, K. Nordström, B. Page, M. Rüfenacht, M. Schönherr, G. Watt, Eur. Phys. J. C **75**, 132 (2015). DOI 10.1140/epjc/s10052-015-3318-8