`POPxf`: An Exchange Format for Polynomial Observable Predictions

Ilaria Brivio (ed.), Ken Mimasu (ed.), Peter Stangl (ed.),
Anke Biekötter, Ana R. Cueto Gómez, Charlotte Knight, Luca Mantani, Eleonora Rossi,
Alejo N. Rossia, Aleks Smolkovič

[arXiv:2511.17348] & Submitted to SciPost Community Reports (YR5)

22nd Workshop of the LHC Higgs WG & 9th General Meeting of the LHC EFT WG

CERN, 3rd of December 2025

Science and Technology Facilities Council

University of Southampton

Idea

*Create a **common data format to publish SMEFT parameterisations** & provide tools/examples for generating/converting them*

Idea

*Create a **common data format to publish SMEFT parameterisations** & provide tools/examples for generating/ converting them*

Theoretical parameterisations are key inputs to statistical models for interpretations of HEP data

- In the SMEFT, parameterisations are rather simple ⇒ polynomials in the Wilson coefficients (or functions thereof)

Idea

*Create a **common data format to publish SMEFT parameterisations** & provide tools/examples for generating/ converting them*

Theoretical parameterisations are key inputs to statistical models for interpretations of HEP data

- In the SMEFT, parameterisations are rather simple ⇒ polynomials in the Wilson coefficients (or functions thereof)

$$\mathscr{L} = \mathscr{L}_{SM} + \sum_i \frac{C_i}{\Lambda^2} \mathscr{O}_i^{(6)} + \ldots \quad \Rightarrow \quad \mathscr{A} = \mathscr{A}_{SM} + \sum_i \frac{C_i}{\Lambda^2} \mathscr{A}_i^{(6)} + \ldots$$

Idea

*Create a **common data format to publish SMEFT parameterisations** & provide tools/examples for generating/ converting them*

Theoretical parameterisations are key inputs to statistical models for interpretations of HEP data

- In the SMEFT, parameterisations are rather simple ⇒ polynomials in the Wilson coefficients (or functions thereof)

$$\mathscr{L} = \mathscr{L}_{SM} + \sum_i \frac{C_i}{\Lambda^2}\mathscr{O}_i^{(6)} + \ldots \quad \Rightarrow \quad \mathscr{A} = \mathscr{A}_{SM} + \sum_i \frac{C_i}{\Lambda^2}\mathscr{A}_i^{(6)} + \ldots$$

$$\sigma \propto |\mathscr{A}|^2 = |\mathscr{A}_{SM}|^2 + \sum_i \frac{C_i}{\Lambda^2}\mathrm{Re}[\mathscr{A}_{SM}^{*}\mathscr{A}_i^{(6)}] + \sum_{i,j} \frac{C_i C_j^{*}}{\Lambda^4}\mathrm{Re}[\mathscr{A}_i^{(6)}\mathscr{A}_j^{(6)*}] + \ldots$$

Idea

*Create a **common data format to publish SMEFT parameterisations** & provide tools/examples for generating/ converting them*

Theoretical parameterisations are key inputs to statistical models for interpretations of HEP data

- In the SMEFT, parameterisations are rather simple ⇒ polynomials in the Wilson coefficients (or functions thereof)

$$\mathscr{L} = \mathscr{L}_{SM} + \sum_i \frac{C_i}{\Lambda^2}\mathcal{O}_i^{(6)} + \ldots \quad \Rightarrow \quad \mathscr{A} = \mathscr{A}_{SM} + \sum_i \frac{C_i}{\Lambda^2}\mathscr{A}_i^{(6)} + \ldots$$

$$\sigma \propto |\mathscr{A}|^2 = |\mathscr{A}_{SM}|^2 + \sum_i \frac{C_i}{\Lambda^2}\mathrm{Re}[\mathscr{A}_{SM}^* \mathscr{A}_i^{(6)}] + \sum_{i,j} \frac{C_i C_j^*}{\Lambda^4}\mathrm{Re}[\mathscr{A}_i^{(6)} \mathscr{A}_j^{(6)*}] + \ldots$$

$1 + n(n+3)/2$ monomials for n Wilson coefficients

Idea

*Create a **common data format to publish SMEFT parameterisations** & provide tools/examples for generating/ converting them*

Theoretical parameterisations are key inputs to statistical models for interpretations of HEP data

- In the SMEFT, parameterisations are rather simple ⇒ polynomials in the Wilson coefficients (or functions thereof)

$$\mathscr{L} = \mathscr{L}_{SM} + \sum_i \frac{C_i}{\Lambda^2}\mathscr{O}_i^{(6)} + \ldots \quad \Rightarrow \quad \mathscr{A} = \mathscr{A}_{SM} + \sum_i \frac{C_i}{\Lambda^2}\mathscr{A}_i^{(6)} + \ldots$$

$$\sigma \propto |\mathscr{A}|^2 = |\mathscr{A}_{SM}|^2 + \sum_i \frac{C_i}{\Lambda^2}\mathrm{Re}[\mathscr{A}_{SM}^* \mathscr{A}_i^{(6)}] + \sum_{i,j} \frac{C_i C_j^*}{\Lambda^4}\mathrm{Re}[\mathscr{A}_i^{(6)} \mathscr{A}_j^{(6)*}] + \ldots$$

$1 + n(n+3)/2$ monomials for n Wilson coefficients

Or functions of polynomials, *e.g.*, $BR_{i\to f} \equiv \Gamma_{i\to f}/\Gamma_{\text{tot.}}$

Current status

SMEFT parameterizations can be quite challenging to obtain

- Large-scale MC simulation: reweighting, templates,…
- Including higher order effects increases computational load

Current status

SMEFT parameterizations can be quite challenging to obtain

- Large-scale MC simulation: reweighting, templates,…
- Including higher order effects increases computational load

Experimental & theoretical groups often derive their own numbers

- Large duplication of effort
- Time consuming & potential for human error
- Important but currently not easy to compare & validate calculations

Current status

SMEFT parameterizations can be quite challenging to obtain

- Large-scale MC simulation: reweighting, templates,…
- Including higher order effects increases computational load

Experimental & theoretical groups often derive their own numbers

- Large duplication of effort
- Time consuming & potential for human error
- Important but currently not easy to compare & validate calculations

Lots of exp/th interactions: technical expertise required

- Support for existing tools, *e.g.*, **`SMEFTsim`**, **`SMEFTatNLO`**,…
- A lot of re-inventing/re-learning the wheel involved
- Some crucial inputs use analytical results, *e.g.*, $h \to \gamma\gamma$

`POPxf`: what it does

1) Stores the monomial coefficients of a generic polynomial

2) Includes uncertainties & correlation information

- **MC, PDF, scale,…**

3) Sufficient metadata to ensure reproducibility

- **Assumptions, settings, code versions, links to documentation**

`POPxf`: what it does

1) Stores the monomial coefficients of a generic polynomial

2) Includes uncertainties & correlation information

- **MC, PDF, scale,…**

3) Sufficient metadata to ensure reproducibility

- **Assumptions, settings, code versions, links to documentation**

Can help to address the aforementioned challenges

- Re-usable, community-validated parameterisations
- State of the art theory: analytic results, higher orders, latest tools
- State of the art experiment: acceptance effects, tools e.g. **`EFT2Obs`**

➡ Best predictions once-and-for-all: improved interpretations of HEP data

`POPxf`: what it does

1) Stores the monomial coefficients of a generic polynomial

2) Includes uncertainties & correlation information

- **MC, PDF, scale,…**

3) Sufficient metadata to ensure reproducibility

- **Assumptions, settings, code versions, links to documentation**

Can help to address the aforementioned challenges

- Re-usable, community-validated parameterisations
- State of the art theory: analytic results, higher orders, latest tools
- State of the art experiment: acceptance effects, tools e.g. **`EFT2Obs`**

➡ Best predictions once-and-for-all: improved interpretations of HEP data

Applies to a generic polynomial, not limited to SMEFT

- Additional parametric dependences can be encoded, *e.g.*, nuisance parameters

Formalism

A **`POPxf`** file encodes a set of M observable predictions, O_m

- The O_m depend on a basis of S parameters, C_s
- …via K polynomials, P_k, each characterized by polynomial coefficients, $\vec{p}_k$
- Predictions are given by observable expressions, $E_m \Rightarrow O_m = E_m(P_1, P_2, \ldots, P_K)$

Formalism

A **`POPxf`** file encodes a set of M observable predictions, O_m

- The O_m depend on a basis of S parameters, C_s
- …via K polynomials, P_k, each characterized by polynomial coefficients, $\vec{p}_k$
- Predictions are given by observable expressions, $E_m \Rightarrow O_m = E_m(P_1, P_2, \ldots, P_K)$

Each polynomial can be expressed as a scalar product between $\vec{p}_k$ and a vector of parameter monomials, $\vec{V}$

$$P_k = \vec{p}_K \cdot \vec{V}$$
$$= a_k + \vec{b}_k \cdot \vec{C} + \vec{c}_k \cdot \mathrm{vech}(\vec{C} \otimes \vec{C})$$

Formalism

A **`POPxf`** file encodes a set of M observable predictions, O_m

- The O_m depend on a basis of S parameters, C_s
- …via K polynomials, P_k, each characterized by polynomial coefficients, $\vec{p}_k$
- Predictions are given by observable expressions, $E_m \Rightarrow O_m = E_m(P_1, P_2, \ldots, P_K)$

Each polynomial can be expressed as a scalar product between $\vec{p}_k$ and a vector of parameter monomials, $\vec{V}$

$$P_k = \vec{p}_K \cdot \vec{V}$$
$$= a_k + \vec{b}_k \cdot \vec{C} + \vec{c}_k \cdot \mathrm{vech}(\vec{C} \otimes \vec{C})$$

e.g. for a quadratic polynomial $\vec{V} = \begin{pmatrix} 1 \\ \vec{C} \\ \mathrm{vech}(\vec{C} \otimes \vec{C}) \end{pmatrix}$

Formalism

A **`POPxf`** file encodes a set of M observable predictions, O_m

- The O_m depend on a basis of S parameters, C_s
- …via K polynomials, P_k, each characterized by polynomial coefficients, $\vec{p}_k$
- Predictions are given by observable expressions, $E_m \Rightarrow O_m = E_m(P_1, P_2, \ldots, P_K)$

Each polynomial can be expressed as a scalar product between $\vec{p}_k$ and a vector of parameter monomials, $\vec{V}$

$$P_k = \vec{p}_K \cdot \vec{V}$$
$$= a_k + \vec{b}_k \cdot \vec{C} + \vec{c}_k \cdot \mathrm{vech}(\vec{C} \otimes \vec{C})$$

e.g. for a quadratic polynomial $\vec{V} = \begin{pmatrix} 1 \\ \vec{C} \\ \mathrm{vech}(\vec{C} \otimes \vec{C}) \end{pmatrix}$

- Numerical values of $\vec{p}_k$ uniquely define each polynomial as functions of C_s
- C_s could be real & imaginary components of SMEFT Wilson coefficients

Use modes

In many cases O_m is itself a polynomial ($E_m = P_m$)

- *e.g.*, if O_m are cross sections
- Common enough to have its own special treatment

Use modes

In many cases O_m is itself a polynomial ($E_m = P_m$)

- *e.g.*, if O_m are cross sections
- Common enough to have its own special treatment

Defined two use modes for the **`POPxf`** format

- Single polynomial (SP) and function-of-polynomials (FOP) modes
- SP dedicated to the case of observables as exact polynomials
- In SP the polynomial coefficients are denoted as observable coefficients, $\vec{o}_m$

Use modes

In many cases O_m is itself a polynomial ($E_m = P_m$)

- *e.g.*, if O_m are cross sections
- Common enough to have its own special treatment

Defined two use modes for the **`POPxf`** format

- Single polynomial (SP) and function-of-polynomials (FOP) modes
- SP dedicated to the case of observables as exact polynomials
- In SP the polynomial coefficients are denoted as observable coefficients, $\vec{o}_m$
- FOP predictions can be reduced to SP by Taylor expansion assuming $C_s \ll 1$
- $\vec{o}_m$ of expanded FOP predictions are intended only for evaluating uncertainties
- Allows for identical treatment of uncertainties in both modes (see backup)

`POPxf` format

Use JSON format to serialize polynomial data

- Machine & human readable
- Defined a JSON schema to simplify validation (json.schemastore.org/popxf-1.0.json)
- Correlations must be specified in separate files

`POPxf` format

Use JSON format to serialize polynomial data

- Machine & human readable
- Defined a JSON schema to simplify validation (json.schemastore.org/popxf-1.0.json)
- Correlations must be specified in separate files

Github page www.github.com/pop-xf with several repos

- **json-schemas** : schema files containing documentation from Section 4 of note
- **parser** : python classes that read-in files & command line validator *work in progress*
- **examples** : complete examples of various use modes

`POPxf` format

Use JSON format to serialize polynomial data

- Machine & human readable
- Defined a JSON schema to simplify validation (json.schemastore.org/popxf-1.0.json)
- Correlations must be specified in separate files

Github page www.github.com/pop-xf with several repos

- **json-schemas** : schema files containing documentation from Section 4 of note
- **parser** : python classes that read-in files & command line validator *work in progress*
- **examples** : complete examples of various use modes

Top level fields

- **$schema** : "https://json.schemastore.org/popxf-1.0.json"
- **metadata** : contextual & structural information to interpret and reproduce predictions
- **data** : numerical values for polynomial coefficients & uncertainties

metadata

- O_m – **`observable_names`** (***required***): array of observable names;
- C_s – **`parameters`** (***required***): array of model parameters (e.g. names of Wilson coefficients);
- – **`basis`** (***required***): definition of the parameter basis, e.g. the operator basis in an EFT;
- $C_s(\mu)$ – **`scale`** (***required***): renormalisation scale at which the parameters are defined;
- – **`polynomial_names`** (***required* in FOP mode**): array of polynomial names;
- – **`observable_expressions`** (***required* in FOP mode**): Python-like expressions defining how each observable is computed from the named polynomials;
- – **`polynomial_degree`** (***optional***): the order of truncation in the parameter expansion (default: 2);
- – **`reproducibility`** (***optional***): information needed to reproduce the predictions. Array in which each element may include the following fields:
  - – **`inputs`** (***optional***): numerical values of input parameters that have been used to compute the polynomial coefficients, with optional uncertainties and correlations;
  - – **`tool`** (***optional***): information about the tools/methods used to generate the predictions;
  - – **`description`** (***optional***): text field with a summary of how the predictions were obtained.
- – **`misc`** (***optional***): free-form documentation or provenance metadata.

data

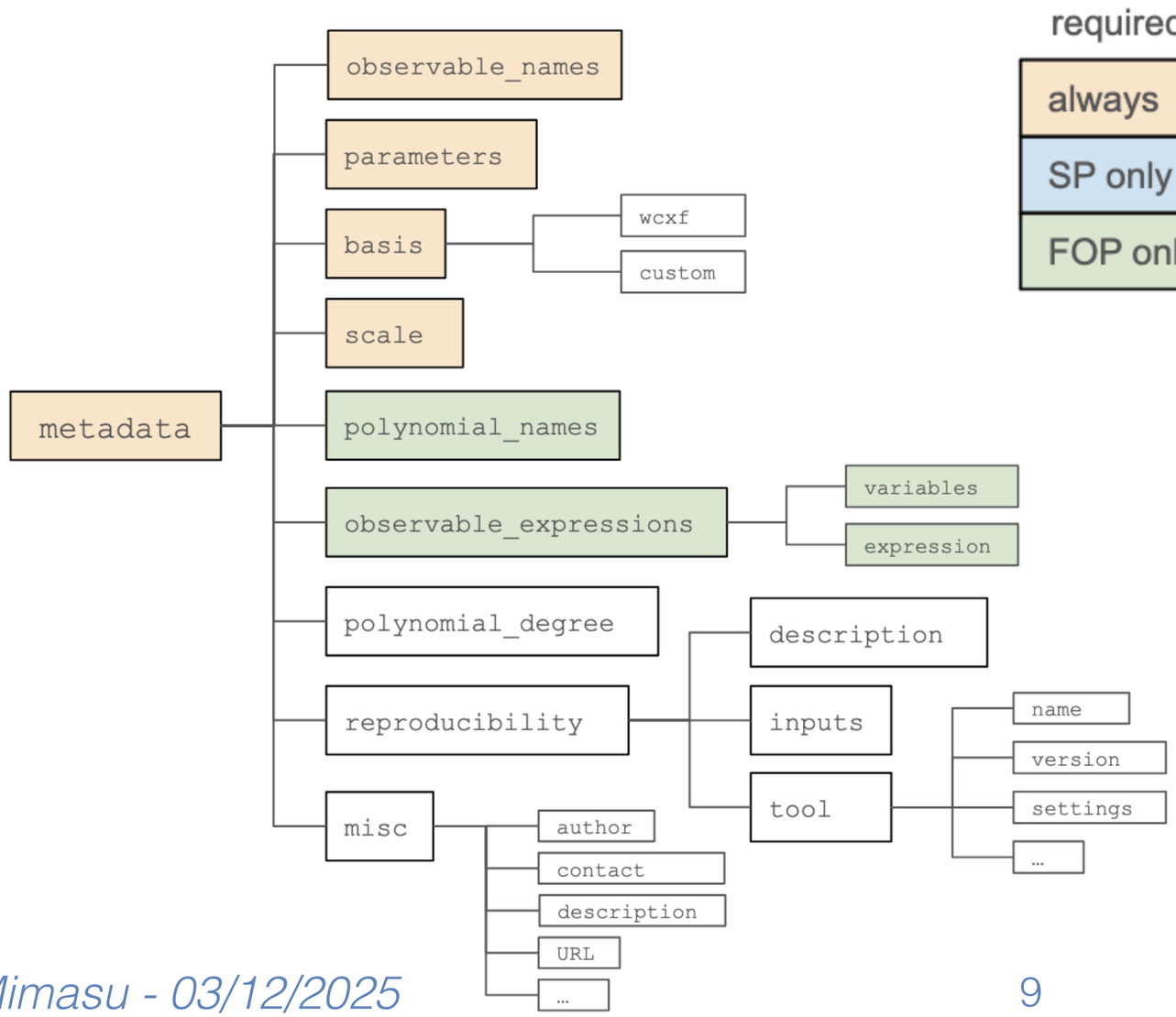
$\vec{p}_k$ – **`polynomial_central`** (***required* in FOP mode**): central values of polynomial coefficients $\vec{p}_k$ for each named polynomial P_k.

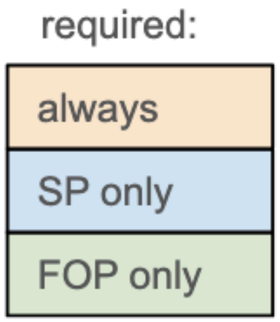
$\vec{o}_m$ – **`observable_central`** (***required* in SP mode; *optional* in FOP mode**): central values of *observable coefficients* $\vec{o}_m$, the polynomial coefficients for each observable O_m.

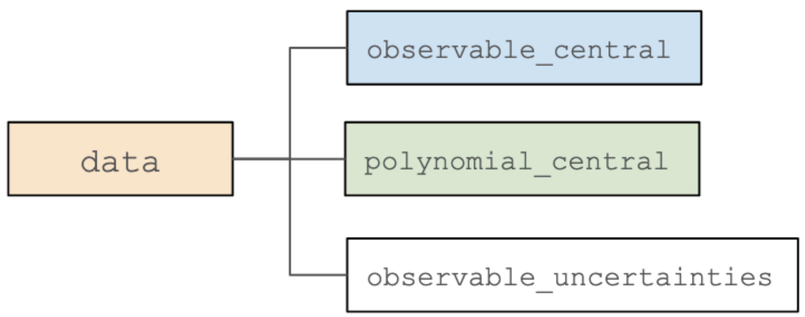
$\vec{\sigma}_m$ – **`observable_uncertainties`** (***optional***): uncertainties $\vec{\sigma}_m$ on the observable coefficients.

data

- $\vec{p}_k$ – **`polynomial_central`** (***required* in FOP mode**): central values of polynomial coefficients $\vec{p}_k$ for each named polynomial P_k.
- $\vec{o}_m$ – **`observable_central`** (***required* in SP mode; *optional* in FOP mode**): central values of *observable coefficients* $\vec{o}_m$, the polynomial coefficients for each observable O_m.
- $\vec{\sigma}_m$ – **`observable_uncertainties`** (***optional***): uncertainties $\vec{\sigma}_m$ on the observable coefficients.







Reproducibility

Optional (but highly recommended) field

- Anything deemed necessary to reproduce the computation
- Essential for validation & comparison

Reproducibility

Optional (but highly recommended) field

- Anything deemed necessary to reproduce the computation
- Essential for validation & comparison

Array with JSON objects describing ‘steps’ to get the result

- **`description`** - human readable information
- **`inputs`** - parameters with errors & correlations, Gaussian or arbitrary PDFs
- **`tool`** - details of the numerical/analytic methods employed
  - **`version`**, documentation, links,…
  - **`settings`**
    - PDFs, scale choice, phase space cuts,…
    - Conventions: schemes, approximations,…
- **`misc`** - free form additional information

Polynomial data

- `polynomial_central` $(\vec{p}_k)$
- `observable_central` $(\vec{o}_m)$
- `observable_uncertainties` $(\vec{\sigma}_m)$

Each component of the polynomial data labeled by a monomial key

- "Stringified" tuple of model parameters with length equal to **`polynomial_degree`**
- Optional real/imaginary tag to indicate which part (defaults to real)

Polynomial data

- `polynomial_central` $(\vec{p}_k)$
- `observable_central` $(\vec{o}_m)$
- `observable_uncertainties` $(\vec{\sigma}_m)$

Each component of the polynomial data labeled by a monomial key

- "Stringified" tuple of model parameters with length equal to **`polynomial_degree`**
- Optional real/imaginary tag to indicate which part (defaults to real)

parameter-independent (SM)	linear	quadratic
`"('','')"`	`"('','C1')"`	`"('C1','C2')"`
`"('','','RR')"`	`"('','C1','RR')"`	`"('C1','C2','RI')"`
	`"('','C1','RI')"`	`"('C1','C2','II')"`

Polynomial data

- `polynomial_central` $(\vec{p}_k)$
- `observable_central` $(\vec{o}_m)$
- `observable_uncertainties` $(\vec{\sigma}_m)$

Each component of the polynomial data labeled by a monomial key

- "Stringified" tuple of model parameters with length equal to **`polynomial_degree`**
- Optional real/imaginary tag to indicate which part (defaults to real)

parameter-independent (SM)	linear	quadratic
`"('','')"`	`"('','C1')"`	`"('C1','C2')"`
`"('','','RR')"`	`"('','C1','RR')"`	`"('C1','C2','RI')"`
	`"('','C1','RI')"`	`"('C1','C2','II')"`

Values are arrays of numbers

```
"observable_central": {
  "('', '')": [1.0, 1.1, 2.3],
  "('', 'C1')": [1.2, 1.3, 0.3],
  "('C1', 'C2')": [1.4, 1.5, 0.7],
  "('C1', 'C3')": [1.6, 1.7, 0.5]
}
```

$\Leftrightarrow$

$$O_1 = 1.0 + 1.2\,C_1 + 1.4\,C_1C_2 + 1.6\,C_1C_3\,,$$
$$O_2 = 1.1 + 1.3\,C_1 + 1.5\,C_1C_2 + 1.7\,C_1C_3\,,$$
$$O_3 = 2.3 + 0.3\,C_1 + 0.7\,C_1C_2 + 0.5\,C_1C_3\,.$$

FOP example: $R_{ij} \equiv \Gamma(W \to \ell_i \nu_i)/\Gamma(W \to \ell_j \nu_j)$

WCxf basis specification

polynomial expressions

```
"metadata": {
  "observable_names": ["Rmue(W->lnu)", "Rtaue(W->lnu)", "Rtaumu(W->lnu)"],
  "parameters": ["phil3_11", "phil3_22", "phil3_33"],
  "basis": {
    "wcxf": {
      "eft": "SMEFT",
      "basis": "Warsaw",
      "sectors": ["dB=de=dmu=dtau=0"]
    }
  },
  "polynomial_names": ["Gamma(W->enu)", "Gamma(W->munu)", "Gamma(W->taunu)"],
  "observable_expressions": [
    {
      "expression": "num / den",
      "variables": {"num": "Gamma(W->munu)", "den": "Gamma(W->enu)"}
    },
    {
      "expression": "num / den",
      "variables": {"num": "Gamma(W->taunu)", "den": "Gamma(W->enu)"}
    },
    {
      "expression": "num / den",
      "variables": {"num": "Gamma(W->taunu)", "den": "Gamma(W->munu)"}
    }
  ],
  "scale": 80.387,
  "reproducibility": [{"tool": {"name": "flavio", "version": "2.6.2"}}],
  "misc": {
    "description": "Using the (alpha, G_F, m_Z) input scheme.",
    "author": ["A. Smolkovic", "P. Stangl"]
  }
},
```

FOP example: $R_{ij} \equiv \Gamma(W \to \ell_i \nu_i)/\Gamma(W \to \ell_j \nu_j)$ continued...


```
"data": {
  "polynomial_central": {
    "('', '')": [0.227, 0.227, 0.227],
    "('', 'phi3_11')": [7737.419, -19812.903, -19812.903],
    "('', 'phi3_22')": [-19812.903, 7737.419, -19812.903],
    "('', 'phi3_33')": [0, 0, 27550.322],
    "('phi3_11', 'phi3_11')": [929672417.073, 1295705157.301, 1295705157.301],
    "('phi3_11', 'phi3_22')": [1390270692.689, 1390270692.689, 2591410314.602],
    "('phi3_11', 'phi3_33')": [0, 0, -1201139621.914],
    "('phi3_22', 'phi3_22')": [1295705157.301, 929672417.073, 1295705157.301],
    "('phi3_22', 'phi3_33')": [0, 0, -1201139621.914],
    "('phi3_33', 'phi3_33')": [0, 0, 835106881.686]
  }
}
```

SP example: $\Gamma(W \to \ell\nu)$


```
"metadata": {
  "basis": {
    "custom": {
      "eft": "SMEFT",
      "basis": "SMEFTatNLO",
      "definition": "https://feynrules.irmp.ucl.ac.be/wiki/SMEFTatNLO"
    }
  },
  "scale": 80.387,
  "parameters": [ "c3pl1", "c3pl2", "cll" ],
  "observable_names": ["Gamma(W -> mu nu_m)"],
  "reproducibility": [{
    "description": "Fixed-order Monte Carlo computation.",
    "tool": {
      "name": "MadGraph5_aMC@NLO",
      "version": "3.4.1",
      "settings":{ "UFO":"SMEFTatNLO 1.0.2",  "perturbative_order":"LO" }
    },
    "inputs":{ "m_W": 80.387, "G_F": 1.1663787e-5, "m_Z": 91.1876, "Lambda":1000 }
  }],
  "misc":{
    "description":"W-boson partial width to mu nu_m [GeV], (m_W,G_F,m_Z) scheme",
    "author": ["E. Celada", "L. Mantani", "K. Mimasu"],
    "contact": "ken.mimasu@soton.ac.uk"
  }
},
```

SP example: $\Gamma(W \to \ell\nu)$ continued…


```
"data": {
  "observable_central": {
     "('', '')": [0.22729],
     "('', 'c3pl1')": [-0.0137796],
     "('', 'c3pl2')": [0.0137786],
     "('', 'cll')": [0.0137796],
     "('c3pl1', 'c3pl1')": [0.000208845],
     "('c3pl2', 'c3pl2')": [0.00020885],
     "('cll', 'cll')": [0.00020884],
     "('c3pl1', 'c3pl2')": [-0.00041769],
     "('c3pl1', 'cll')": [-0.00041768],
     "('c3pl2', 'cll')": [0.00041768]
}
```

Example: uncertainties & correlations

total, parameter dependent

```
"observable_uncertainties": {
  "total": {
    "('', '')": [0.05, 0.06, 0.01],
    "('', 'C1')": [0.1, 0.12, 0.01],
    "('C1', 'C2')": [0.02, 0.03, 0.02],
    "('C1', 'C3')": [0.05, 0.06, 0.01]
  }
}
```

breakdown

```
"observable_uncertainties": {
  "MC_stats": {
    "('', '')": [0.002, 0.0012, 0.001],
    "('', 'C1')": [0.001, 0.0015, 0.0001]
  },
  "scale": {
    "('', '')": [0.04, 0.05, 0.06],
    "('', 'C1')": [0.1, 0.12, 0.01]
  },
  "PDF": {
    "('', '')": [0.03, 0.04, 0.05],
    "('', 'C1')": [0.02, 0.08, 0.01]
  }
}
```

Example: uncertainties & correlations

total, parameter dependent

```
"observable_uncertainties": {
  "total": {
    "('', '')": [0.05, 0.06, 0.01],
    "('', 'C1')": [0.1, 0.12, 0.01],
    "('C1', 'C2')": [0.02, 0.03, 0.02],
    "('C1', 'C3')": [0.05, 0.06, 0.01]
  }
}
```

breakdown

```
"observable_uncertainties": {
  "MC_stats": {
    "('', '')": [0.002, 0.0012, 0.001],
    "('', 'C1')": [0.001, 0.0015, 0.0001]
  },
  "scale": {
    "('', '')": [0.04, 0.05, 0.06],
    "('', 'C1')": [0.1, 0.12, 0.01]
  },
  "PDF": {
    "('', '')": [0.03, 0.04, 0.05],
    "('', 'C1')": [0.02, 0.08, 0.01]
  }
}
```

Sample correlation file

```
{
  "$schema": "https://json.schemastore.org/popxf-corr-1.0.json",
  "593771630098eb5325684131f80b4224": {
    "row_names": ["BR(Bs->mumu)", "BR(B0->mumu)"],
    "col_names": ["BR(Bs->mumu)", "BR(B0->mumu)"],
    "correlations": {
      "total":[
        [1.0, 0.407],
        [0.407, 1.0]
      ]
    }
  }
}
```

parameter-independent (SM) correlations

- Also has a JSON schema
- Many observables x many parameters
- Potential for very large files
- Alternative hdf5 format serialization for correlations

User-friendliness

```
(syspy3.11) LQPJQJXFNX:parser ken$ bin/validate-popxf -h
usage: validate-popxf [-h] [-v] [-q] [--show-data] [--show-uncertainties] input_file

Validate and parse POPxf JSON files.

positional arguments:
  input_file            Path to the POPxf JSON file to parse

options:
  -h, --help            show this help message and exit
  -v, --verbose         Print detailed information about the parsed data
  -q, --quiet           Suppress output (only show errors)
  --show-data           Display polynomial data (can be large)
  --show-uncertainties  Display uncertainty information

Examples:
  validate-popxf input.json
  validate-popxf input.json --verbose
  validate-popxf input.json --quiet
  validate-popxf input.json --show-data
```

Command line validation tool

```
(syspy3.11) LQPJQJXFNX:parser ken$ bin/validate-popxf examples/Wlnu.json

======================================================================
Reading JSON file: examples/Wlnu.json
======================================================================
✓ examples/Wlnu.json is valid
======================================================================
POPxfParser Object Properties
======================================================================

Schema version: popxf-1.0.json

Mode: Function-Of-Polynomials (FOP)
Polynomial Degree: 2
Length (number of observables): 3
Observable Names: ['Rmue(W->lnu)', 'Rtaue(W->lnu)', 'Rtaumu(W->lnu)']
Parameters: ['ll_1221', 'phiD', 'phiWB', 'phil3_11', 'phil3_12', 'phil3_13', 'phil3_22', 'phil3_23', 'phil3_33']
Scale: 80.387 [GeV]
```

User-friendliness

```
(syspy3.11) LQPJQJXFNX:parser ken$ bin/validate-popxf -h
usage: validate-popxf [-h] [-v] [-q] [--show-data] [--show-uncertainties] input_file

Validate and parse POPxf JSON files.

positional arguments:
  input_file            Path to the POPxf JSON file to parse

options:
  -h, --help            show this help message and exit
  -v, --verbose         Print detailed information about the parsed data
  -q, --quiet           Suppress output (only show errors)
  --show-data           Display polynomial data (can be large)
  --show-uncertainties  Display uncertainty information

Examples:
  validate-popxf input.json
  validate-popxf input.json --verbose
  validate-popxf input.json --quiet
  validate-popxf input.json --show-data
```

Command line validation tool

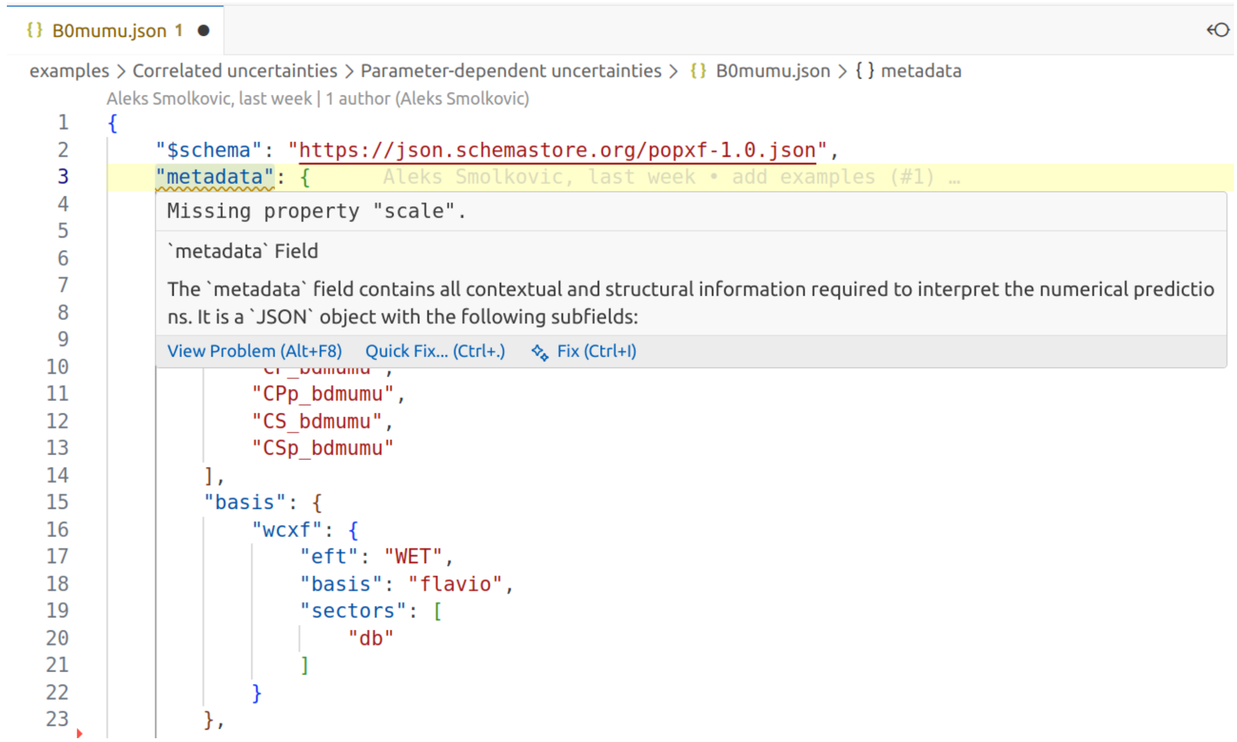
```
(syspy3.11) LQPJQJXFNX:parser ken$ bin/validate-popxf examples/Wlnu.json

======================================================================
Reading JSON file: examples/Wlnu.json
======================================================================
✓ examples/Wlnu.json is valid
======================================================================
POPxfParser Object Properties
======================================================================

Schema version: popxf-1.0.json

Mode: Function-Of-Polynomials (FOP)
Polynomial Degree: 2
Length (number of observables): 3
Observable Names: ['Rmue(W->lnu)', 'Rtaue(W->lnu)', 'Rtaumu(W->lnu)']
Parameters: ['ll_1221', 'phiD', 'phiWB', 'phil3_11', 'phil3_12', 'phil3_13', 'phil3_22', 'phil3_23', 'phil3_33']
Scale: 80.387 [GeV]
```

JSON schema ⇒ tooltips for standard IDEs



Outlook

POPxf will hopefully become a standard for EFT parameterisations

- Now its up to all of us to adopt it! experiment & theory (see Aleks' talk yesterday)

Outlook

POPxf will hopefully become a standard for EFT parameterisations

- Now its up to all of us to adopt it! experiment & theory (see Aleks' talk yesterday)

Obvious use-cases

- Observables with standardised theoretical definitions or binnings
- Higgs STXS (activity in WG2), EWPOs *[Mildner; 2412.07651]*, analytic results

Outlook

POPxf will hopefully become a standard for EFT parameterisations

- Now its up to all of us to adopt it! experiment & theory (see Aleks' talk yesterday)

Obvious use-cases

- Observables with standardised theoretical definitions or binnings
- Higgs STXS (activity in WG2), EWPOs *[Mildner; 2412.07651]*, analytic results

https://github.com/pop-xf/parser

- Final structure of the validations/parsing tools is not fixed
- More functionality planned
- Evaluating predictions, comparing files, constructing covariance matrices,…
- Conversions to/from other formats can be built from here (fitting groups, **`EFT2Obs`**,…)
- Contributions welcome!

Backup

Scale

$$O_m = a_m + \vec{C}(\mu) \cdot \vec{b}_m(\mu) + \dots$$

Common scale, μ, at which Wilson coefficients are defined

- $\vec{b}_m(\mu)$: observable is scale-independent up to higher orders

Relevant when RG effects are taken into account

- Important to document how this was done in **`reproducibility`**
- If no running is included, set the scale to the scale of the observable, *e.g.*, m_h, m_Z,…

Can be an array for multiple observables

```
"scale": 91.1876
```
common scale

```
"scale": [100.0, 200.0, 300.0, 400.0, 500.0]
```
distribution with no running included

- *restricted usage of arrays in FOP mode

Uncertainties

for details see [POPxf; 2511.17348] & [Altmanshofer & Stangl; 2103.13370]

Consider Gaussian uncertainties in the O_m

- Can be correlated and depend on the parameters
- Uncertainties on $[\vec{o}_m]_\alpha \Rightarrow [\vec{\sigma}_m]_\alpha$ and correlation matrix $[\rho_{mm'}]_{\alpha\alpha'}$

m : *observables* α : *parameter monomials* $(1, C_1, C_2, C_1C_2, \ldots)$

Correlations can be given across different POPxf files (n)

- $[\vec{\sigma}_m^{(n)}]_\alpha$ and correlation matrix $[\rho_{mm'}^{(nn')}]_{\alpha\alpha'} \Rightarrow \left[\tilde{\Sigma}_{mm'}^{(nn')}\right]_{\alpha\alpha'} = \left[\vec{\sigma}_m^{(n)}\right]_\alpha \left[\rho_{mm'}^{(nn')}\right]_{\alpha\alpha'} \left[\vec{\sigma}_{m'}^{(n)'}\right]_{\alpha'}$

parameter-dependent *parameter-independent*

$$\left[\Sigma^{(nn')}\right]_{mm'} = \sum_{\alpha,\alpha'} \left[\vec{V}(n)\right]_\alpha \left[\tilde{\Sigma}_{mm'}^{(nn')}\right]_{\alpha\alpha'} \left[\vec{V}(n')\right]_{\alpha'} \supset \left[\tilde{\Sigma}_{mm'}^{(nn')}\right]_{00}$$

- Can also specify a breakdown into different sources, *e.g.,* PDF, scale, …