Fiteft manual

Contents

Fieft (**Fit** **e**ffective **f**ield **t**heory) is a tool written entirely in Python with the purpose of computing an approximate likelihood function, the following requirements must be met:

- Python 3.X
- numpy
- pandas
- regex

Along with these mandatory libraries, another library that can be used to assist in minimization is:

- scipy

1 Program structure

Fiteft source code is written in `Fiteft.py` file where we initialize the `fiteft` object, define different likelihood functions and their derivative. Different `likelihood()` functions in `fieft` object will paste numbers into `function.likelihood()` function to do the actual calculation. The graph below is the file structure of Fiteft

```
Fiteft
├─ Fiteft.py
│  └─ likelihood()
├─ function.py
│  └─ likelihood()
└─ data/
   └─ <experiment_name>/
      ├─ observable.csv
      ├─ parameterization.csv
      ├─ correlation.csv
      └─ rotate.csv (optional)
```

1.1 Data structure

column

index

production	decay	acceptance	signature	central	+total	-total	+stat	-stat	+syst	-syst	central_SM	+total_SM	-total_SM
gg->H,0jet,pTH<10GeV	H->ZZ->4l	delta H->ZZ->4l	CS*BR_ZZ/BR_ZZSM	5.90	1.50	-1.30	1.30	-1.20	0.70	-0.60	6.60	0.90	-0.90
gg->H,0jet,10<=pTH<200GeV	H->ZZ->4l	delta H->ZZ->4l	CS*BR_ZZ/BR_ZZSM	23.60	3.10	-2.80	2.50	-2.40	1.80	-1.50	20.60	1.60	-1.60
gg->H,1jet,pTH<60GeV	H->ZZ->4l	delta H->ZZ->4l	CS*BR_ZZ/BR_ZZSM	3.70	1.80	-1.80	1.40	-1.40	1.20	-1.20	6.50	0.90	-0.90
gg->H,1jet,60<=pTH<120GeV	H->ZZ->4l	delta H->ZZ->4l	CS*BR_ZZ/BR_ZZSM	4.80	1.30	-1.20	1.10	-1.10	0.60	-0.50	4.50	0.60	-0.60
gg->H,1jet,120<=pTH<200GeV	H->ZZ->4l	delta H->ZZ->4l	CS*BR_ZZ/BR_ZZSM	0.50	0.30	-0.29	0.27	-0.26	0.15	-0.13	0.75	0.13	-0.13
gg->H,>=2jet,mjj<350GeV,pTH<60GeV	H->ZZ->4l	delta H->ZZ->4l	CS*BR_ZZ/BR_ZZSM	0.60	1.30	-1.20	1.20	-1.10	0.50	-0.50	1.17	0.27	-0.27
gg->H,>=2jet,mjj<350GeV,60<=pTH<120GeV	H->ZZ->4l	delta H->ZZ->4l	CS*BR_ZZ/BR_ZZSM	0.40	1.00	-1.00	0.80	-0.80	0.50	-0.50	1.80	0.40	-0.40
gg->H,>=2jet,mjj<350GeV,120<=pTH<200GeV	H->ZZ->4l	delta H->ZZ->4l	CS*BR_ZZ/BR_ZZSM	0.50	0.40	-0.40	0.40	-0.30	0.20	-0.20	0.94	0.21	-0.21
gg->H,>=2jet,350<=mjj<700GeV,pTH<200GeV	H->ZZ->4l	delta H->ZZ->4l	CS*BR_ZZ/BR_ZZSM	1.70	0.70	-0.60	0.60	-0.60	0.30	-0.30	0.61	0.13	-0.13
gg->H,>=2jet,mjj>=700GeV,pTH<200GeV	H->ZZ->4l	delta H->ZZ->4l	CS*BR_ZZ/BR_ZZSM	0.20	0.40	-0.40	0.40	-0.30	0.20	-0.20	0.27	0.06	-0.06

Figure 1: An example of a `observable.csv` file, representing the first 10 measurement results of the experiment ATLAS-2021-053[1].

The Fiteft database is stored in the directory

```
Fiteft/data/{experiment_name}
```

and consists of three main files:

- **observable.csv** : This file contains data on physical observables and errors provided by the experiment.
- **correlation.csv** : This file contains information on the correlation matrix provided by the experiment.
- **parametrization.csv** : This file contains information on the parameter matrix, which can be provided by the experimental paper or by the user for any model.

Other auxiliary files may also be present, and the program can run without these files. They provide additional information to help improve the data fitting results.

- **correlation_theory.csv** : This file contains information on the theoretical correlation matrix.
- **rotation.csv** : A basis transformation matrix that converts from basis c to basis c' with fewer basis vectors, helping reduce the number of parameters during data fitting.

1.1.1 observable.csv

This data file has two important parts:

index : Five important columns are required:

- production: Name of the Higgs boson production channels in stage 1.2. Names can be arbitrary, either letters or numbers. The name `production+decay` in one row must not match `production+decay` in another row.

- decay: Name of the Higgs boson decay channels for the corresponding bin. Names can be arbitrary, either letters or numbers. The name `production+decay` in one row must not match `production+decay` in another row.
- acceptance: Type of acceptance used for the bin corresponding to the `decay` channel. Names can be arbitrary and should not match `decay`. A simple naming convention is "{...}`decay`" where {...} can be any character(s).
- signature: Characteristics of the experimental result of the production+decay channel. We allow some specific characteristics listed in 1.

column : The following information is needed:

- central: Central value
- +total: Total error on the right side
- –total: Total error on the left side

Predicted values from the SM depend on the signature of the measurement results, and these predictions are listed in 1

- central_SM: SM theoretical prediction value

Extra data that help improve the fitting procedure:

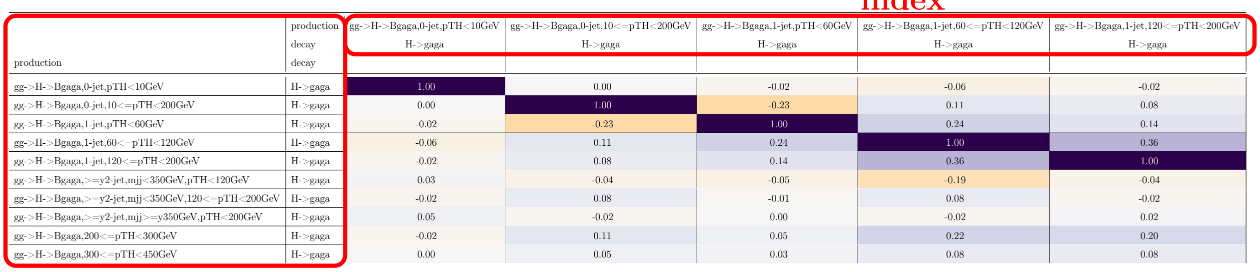
- +syst: Systematic error on the right side
- –syst: Systematic error on the left side
- +stat: Statistical error on the right side
- –stat: Statistical error on the left side
- –total_SM: SM theoretical prediction total error on the left side
- +total_SM: SM theoretical prediction total error on the right side

1.1.2 correlation.csv

Figure 2 is an example of the `correlation.csv` file, which contains data on the correlation between any two bins. Each bin is a combination of a Higgs boson production channel and a Higgs boson decay channel. It is easy to see that the correlation of a bin with itself is 1, representing the diagonal of the matrix. This must be a square matrix with the number of rows and columns equal to the number of measured physical observables in the `obersvable.csv` file. Each bin is represented by a combination of the `production` and `decay`, names, and each combination must correspond to a combination in the `observable.csv`file.

Physical quantity	Signature	observable predicted by the SM
$(\sigma_i \times B_j)$	CS*BR	$(\sigma_i \times B_j)_{SM}$
$(\sigma_i \times B_j)$normalized to SM Signal strength	CS*BR/(CS_SM*BR_SM)	1
$(\sigma_i \times B_{\mathrm{Z}})/B_{\mathrm{Z}}^{SM}$	CS*BR_ZZ/BR_ZZSM	σ_i^{SM}
$B_i/B_{\mathrm{H}\to\mathrm{ZZ}}$	BR/BR_ZZ	$(B_i/B_{\mathrm{Z}})_{SM}$

Table 1: Here are the currently supported signature in Fiteft, the names to be filled in the signature column in the observable.csv file, and the additional predicted values that need to be provided by the user.



index$^{\top}$

index

production	production decay decay	gg->H->Bgaga,0-jet,pTH<10GeV H->gaga	gg->H->Bgaga,0-jet,10<=pTH<200GeV H->gaga	gg->H->Bgaga,1-jet,pTH<60GeV H->gaga	gg->H->Bgaga,1-jet,60<=pTH<120GeV H->gaga	gg->H->Bgaga,1-jet,120<=pTH<200GeV H->gaga
gg->H->Bgaga,0-jet,pTH<10GeV	H->gaga	1.00	0.00	-0.02	-0.06	-0.02
gg->H->Bgaga,0-jet,10<=pTH<200GeV	H->gaga	0.00	1.00	-0.23	0.11	0.08
gg->H->Bgaga,1-jet,pTH<60GeV	H->gaga	-0.02	-0.23	1.00	0.24	0.14
gg->H->Bgaga,1-jet,60<=pTH<120GeV	H->gaga	-0.06	0.11	0.24	1.00	0.36
gg->H->Bgaga,1-jet,120<=pTH<200GeV	H->gaga	-0.02	0.08	0.14	0.36	1.00
gg->H->Bgaga,>=y2-jet,mjj<350GeV,pTH<120GeV	H->gaga	0.03	-0.04	-0.05	-0.19	-0.04
gg->H->Bgaga,>=y2-jet,mjj<350GeV,120<=pTH<200GeV	H->gaga	-0.02	0.08	-0.01	0.08	-0.02
gg->H->Bgaga,>=y2-jet,mjj>=y350GeV,pTH<200GeV	H->gaga	0.05	-0.02	0.00	-0.02	0.02
gg->H->Bgaga,200<=pTH<300GeV	H->gaga	-0.02	0.11	0.05	0.22	0.20
gg->H->Bgaga,300<=pTH<450GeV	H->gaga	0.00	0.05	0.03	0.08	0.08

Figure 2: An example of a corelation.csv file, representing the first 10 rows, 5 columns of the experiment ATLAS-CONF-2020-053 [2].

1.1.3 parametrization.csv

Table 2 lists a subset of the parameters from the influence matrix $\mathbf{A}$ of SMEFT parameters, with a total of 34 parameters. In the column labeled `bin`, surrounded by a red rectangle with the name **prod.** i, we need to ensure that we fully list the linear parametrization of **all** production channels `production`. The column **decay** j lists all decay channels `decay`. These are also present in the `acceptance` column in the `observable.csv` file. Another mandatory requirement is the total decay parameterization of the Higgs boson $(\mathbf{a}_{\mathrm{H}}^{\mathrm{decay}})^{\top}$. All the numbers here represent the linear influence of the parameter on the measurement bin in question. For example, for row 1 column 1 of Table 2, , we can infer that:

$$\frac{\sigma^{\mathrm{SMEFT}}(\mathrm{gg}->\mathrm{H}, 0\mathrm{jet}, \mathrm{pTH} < 10\mathrm{GeV})|_{\mathrm{cHbox}=1}}{\sigma^{\mathrm{SM}}(\mathrm{gg}->\mathrm{H}, 0\mathrm{jet}, \mathrm{pTH} < 10\mathrm{GeV})} = 1 + 0.12, \tag{1}$$

bin	cHbox	cHDD	cHG	cuH	cuG	c(3)Hl	c0ll	cHWB	c(1)Hl	cHe	c(1)Hq	c(3)Hq	cHu	cHd	cHW
gg->H->Bgaga,0-jet,pTH<10GeV	0.12	-0.03	42.00	-0.12	1.59	-0.12	0.06	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
gg->H->Bgaga,0-jet,10<=pTH<200GeV	0.12	-0.03	42.20	-0.12	1.62	-0.12	0.06	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
gg->H->Bgaga,1-jet,pTH<60GeV	0.12	-0.03	44.00	-0.13	1.60	-0.13	0.07	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
gg->H->Bgaga,1-jet,60<=pTH<120GeV	0.12	-0.03	43.50	-0.12	1.58	-0.12	0.06	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
gg->H->Bgaga,1-jet,120<=pTH<200GeV	0.12	-0.03	44.00	-0.12	1.60	-0.11	0.06	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
gg->H->Bgaga,>=y2-jet,mjj<350GeV,pTH<120GeV	0.12	-0.03	46.50	-0.13	1.61	-0.13	0.07	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
gg->H->Bgaga,>=y2-jet,mjj<350GeV,120<=pTH<200GeV	0.12	-0.03	46.00	-0.13	1.48	-0.13	0.07	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
gg->H->Bgaga,>=y2-jet,mjj>=y350GeV,pTH<200GeV	0.12	-0.03	46.00	-0.14	1.51	-0.13	0.07	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
gg->H->Bgaga,200<=pTH<300GeV	0.12	-0.03	47.00	-0.12	1.69	-0.12	0.06	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
gg->H->Bgaga,300<=pTH<450GeV	0.12	-0.03	60.00	-0.12	2.10	-0.11	0.06	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
gg->H->Bgaga,pTH>=y450GeV	0.12	-0.03	73.85	-0.12	2.68	-0.12	0.06	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
qq->Hqq->Bgaga,<=1-jet	0.12	-0.01	0.00	0.00	0.00	-0.37	0.18	0.05	0.00	0.00	0.00	0.40	0.03	-0.01	0.21
qq->Hqq->Bgaga,>=y2-jet,mjj<350GeV,VHveto	0.12	-0.01	0.00	0.00	0.00	-0.36	0.18	0.04	0.00	0.00	-0.00	0.01	0.01	-0.00	0.04
qq->Hqq->Bgaga,>=y2-jet,mjj<350GeV,VHtopo	0.12	-0.01	0.00	0.00	0.00	-0.36	0.18	0.10	0.00	0.00	-0.01	2.07	0.15	-0.06	0.64
qq->Hqq->Bgaga,>=y2-jet,350<=mjj<700GeV,pTH<200GeV	0.12	-0.01	0.00	0.00	0.00	-0.37	0.18	0.02	0.00	0.00	0.00	-0.37	-0.02	0.01	-0.14
qq->Hqq->Bgaga,>=y2-jet,mjj>=y700GeV,pTH<200GeV	0.12	-0.01	0.00	0.00	0.00	-0.36	0.18	0.02	0.00	0.00	0.01	-0.38	-0.02	0.01	-0.13
qq->Hqq->Bgaga,>=y2-jet,mjj>=y350GeV,pTH>=y200GeV	0.12	-0.01	0.00	0.00	0.00	-0.36	0.18	0.04	0.00	0.00	0.05	-1.33	-0.10	0.03	0.19
qq->Hlv->Bgaga,pTV<150GeV	0.12	-0.03	0.00	0.00	0.00	-0.24	0.18	0.00	0.00	0.00	0.00	1.43	0.00	0.00	0.86
qq->Hlv->Bgaga,pTV>=y150GeV	0.12	-0.03	0.00	0.00	0.00	-0.24	0.18	0.00	0.00	0.00	0.00	4.44	0.00	0.00	1.07
gg/qq->Hll->Bgaga,pTV<150GeV	0.12	0.01	0.00	0.01	0.04	-0.22	0.17	0.27	-0.04	-0.03	-0.10	1.38	0.34	-0.10	0.61
gg/qq->Hll->Bgaga,pTV>=y150GeV	0.12	0.01	0.00	0.02	0.10	-0.22	0.17	0.27	-0.04	-0.03	-0.45	3.88	1.01	-0.30	0.60
ttH->Bgaga,pTH<60GeV	0.12	-0.03	0.41	-0.12	-0.76	-0.12	0.06	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
ttH->Bgaga,60<=pTH<120GeV	0.12	-0.03	0.45	-0.11	-0.79	-0.12	0.06	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
ttH->Bgaga,120<=pTH<200GeV	0.12	-0.03	0.55	-0.11	-0.89	-0.11	0.06	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
ttH->Bgaga,pTH>=y200GeV	0.12	-0.03	0.73	-0.11	-0.99	-0.11	0.06	-0.00	0.00	0.00	-0.00	0.02	0.00	-0.00	0.00
tH->Bgaga,	0.12	-0.03	0.25	-0.08	-0.12	-0.27	0.13	0.00	0.00	0.00	0.00	0.32	0.00	0.00	0.18
gg->H->BZZ,0-jet,pTH<10GeV	0.12	-0.03	42.00	-0.12	1.59	-0.12	0.06	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
gg->H->BZZ,0-jet,10<=pTH<200GeV	0.12	-0.03	42.20	-0.12	1.62	-0.12	0.06	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
gg->H->BZZ,1-jet,pTH<60GeV	0.12	-0.03	44.00	-0.13	1.60	-0.13	0.07	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
gg->H->BZZ,1-jet,60<=pTH<120GeV	0.12	-0.03	43.50	-0.12	1.58	-0.12	0.06	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
gg->H->BZZ,1-jet,120<=pTH<200GeV	0.12	-0.03	44.00	-0.12	1.60	-0.11	0.06	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
gg->H->BZZ,>=y2-jet,pTH<200GeV	0.12	-0.03	46.30	-0.13	1.56	-0.13	0.07	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
gg->H->BZZ,pTH>=y200GeV	0.12	-0.03	52.07	-0.12	1.90	-0.12	0.06	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
qq->Hqq->BZZ,VBFtopo	0.12	-0.01	0.00	0.00	0.00	-0.36	0.18	0.02	0.00	0.00	0.01	-0.38	-0.02	0.01	-0.13
qq->Hqq->BZZ,>=y2-jet,mjj<350GeV,VHtopo	0.12	-0.01	0.00	0.00	0.00	-0.36	0.18	0.10	0.00	0.00	-0.01	2.07	0.15	-0.06	0.64
qq->Hqq->BZZ,>=y2-jet,mjj>=y350GeV,pTH>=y200GeV	0.12	-0.01	0.00	0.00	0.00	-0.36	0.18	0.04	0.00	0.00	0.05	-1.33	-0.10	0.03	0.19
VHlep->BZZ,	0.12	-0.01	0.00	0.00	0.02	-0.24	0.18	0.12	-0.01	-0.01	-0.05	1.87	0.17	-0.05	0.79
top->BZZ,	0.12	-0.03	0.47	-0.11	-0.74	-0.14	0.07	-0.00	0.00	0.00	-0.00	0.05	0.00	-0.00	0.03
qq->Hlv->Bbb,pTV<250GeV	0.12	-0.03	0.00	0.00	0.00	-0.24	0.18	0.00	0.00	0.00	0.00	1.68	0.00	0.00	0.89
qq->Hlv->Bbb,pTV>=y250GeV	0.12	-0.03	0.00	0.00	0.00	-0.23	0.17	0.00	0.00	0.00	0.00	10.60	0.00	0.00	1.07
gg/qq->Hll->Bbb,pTV<150GeV	0.12	0.01	0.00	0.01	0.04	-0.22	0.17	0.27	-0.04	-0.03	-0.10	1.38	0.34	-0.10	0.61
gg/qq->Hll->Bbb,150<=pTV<250GeV	0.12	0.01	0.00	0.02	0.09	-0.22	0.17	0.26	-0.04	-0.03	-0.22	2.32	0.60	-0.17	0.59
gg/qq->Hll->Bbb,pTV>=y250GeV	0.12	0.02	0.00	0.03	0.12	-0.23	0.17	0.30	-0.04	-0.03	-0.91	7.38	1.91	-0.58	0.63
H->gaga	0.12	-0.24	0.00	0.03	0.00	-0.36	0.18	22.40	0.00	0.00	0.00	0.00	0.00	0.00	-13.08
H->ZZ->4l	0.12	0.01	0.00	0.00	0.00	-0.23	0.18	0.30	0.13	-0.10	0.00	0.00	0.00	0.00	-0.30
H->bb(VH)	0.12	-0.03	0.00	0.00	0.00	-0.12	0.06	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
H->all	0.12	-0.03	1.36	-0.01	0.05	-0.15	0.08	0.05	0.00	0.00	0.00	0.01	0.00	0.00	-0.05
none	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00

prod. i

decay j

decay H

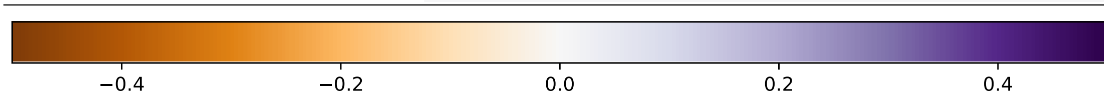


Table 2: An example of a `parametrization.csv`, file, which represents the linear impact of SMEFT parameters on the production and decay channels of the Higgs boson for the ATLAS-CONF-2020-053 [2] experiment. This table includes the impact of 34 SMEFT parameters.

where we only keep the interference term 0.12. From the first row, we can infer that:

$$\frac{\sigma^{\mathrm{SMEFT}}(\mathrm{gg} \rightarrow \mathrm{H}, 0\mathrm{jet}, \mathrm{p_H^T} < 10\mathrm{GeV})}{\sigma^{\mathrm{SM}}(\mathrm{gg} \rightarrow \mathrm{H}, 0\mathrm{jet}, \mathrm{p_H^T} < 10\mathrm{GeV})}$$

$= 1 +$ `0.12 cHbox -0.0294 cHDD+ 42.0 cHG -0.117 cuH+ 1.59 cuG`

```
-0.117 c(3)Hl + 0.0587 c0ll+ 0.0 cHWB+ 0.0 c(1)Hl+ 0.0 cHe+ 0.0 c(1)Hq
+ 0.0 c(3)Hq+ 0.0 cHu+ 0.0 cHd+ 0.0 cHW+ 0.0 cHB+ 0.0 cG
+ 0.0 cuW+ 0.0 cuB+ 0.0 cqq+ 0.0 c(1)qq+ 0.0 c(3)qq
+ 0.0 c(31)qq+ 0.0 cuu+ 0.0 c(1)uu+ 0.0 c(1)ud+ 0.0 c(8)ud + 0.0 c(1)qu
+ 0.0 c(8)qu+ 0.0 c(1)qd+ 0.0 c(8)qd+ 0.0 cW+ 0.0 cdH+ 0.0 ceH.
```

1.1.4 rotate.csv (optional)

The file contains data on the transformation matrix from basis c to basis c', or in other words, the matrix $\overline{\mathbf{M}_{\lambda>a}}$, for example, table **??**. You can choose one of the two tables, whichever you prefer. The use of a rotation matrix is optional, but it may not be possible to fit (very large confidence intervals) for some parameters.

1.2 The fiteft object

1.2.1 Initialize the fiteft

For more information about initializing `fiteft`, refer to the example in subsection ??.

```
fiteft(experiment='ATLAS-CONF-2020-053')
```

Argument

experiment : string
The name of the directory containing the data for an experiment, located in the directory `Fiteft/data`.

Return: fiteft object.
Example

```
>>> from Fiteft import fiteft
>>> a = fiteft(experiment='ATLAS-CONF-2020-053')
Your input to the likelihood function is a DataFrame with one of these colums:
['c(3)Hq', 'c[1]HW-HB-HWB-HDD-uW-uB', 'c[2]HW-HB-HWB-HDD-uW-uB', 'c[3]HW-HB-HWB-HDD-uW-uB', 'c[1]Hu-Hd-Hq(1)', 'c[1]Hl(1)-He', 'c[1]Hl(3)-ll0', 'c[1]HG-uG-uH-top', 'c[2]HG-uG-uH-top', 'c[3]HG-uG-uH-top']
```

1.2.2 Changing attribute of the likelihood function

The user can change the type of likelihood and the method of computing the likelihood through

```
fiteft.attribute.update(dict)
```

Argument: Python dictionary `{key: value}`.

key : `string`, can be a value between `'likelihood_type'` or `'devide'`:

value `likelihood_type`: The type of likelihood can be chosen from the following options:['variable Gaussian 0', 'vg0', 0], ['variable Gaussian 1', 'vg1', 1], ['normal Gaussian', 'ng', 2]. Each of these values serves the same function and represents a different method of computing the likelihood. The specific types of likelihood are explained in detail in Equations (2), (3), (4), respectively. The default setting for this property is `'normal Gaussian'`.

$$
\Delta_{\mathbf{y}} = \begin{cases} (\Delta_{\mathbf{y}}^{+} + \Delta_{\mathbf{y}}^{-})/2, & (2) \\ \sqrt{\dfrac{2\Delta^{+} \circ \Delta^{-}}{\Delta^{+} + \Delta^{-}} + \dfrac{2\Delta^{+} - \Delta^{-}}{\Delta^{+} + \Delta^{-}} \circ (\mathbf{y_c} - \mathbf{y}_{\mathrm{SM}})}, & (3) \\ \sqrt{\Delta^{+} \circ \Delta^{-} + (\Delta^{+} - \Delta^{-}) \circ (\mathbf{y_c} - \mathbf{y}_{\mathrm{SM}})}, & (4) \end{cases}
$$

`devide`: The user can choose between `True` and `False` to represent nonlinear and linear parameterization, respectively. Both choices correspond to two different methods of theoretical prediction parameterization described in Equations (5), (6). The default setting for this property is `False`, which corresponds to Equation (6).

$$
\mathbf{y}^{\mathrm{SMEFT}}(\mathbf{c}) = \begin{cases} \mathbf{y}_{\mathrm{SM}} \circ (1 + \mathbf{N}.\mathbf{c})/(1 + \mathbf{D}.\mathbf{c}) & (5) \\ \mathbf{y}_{\mathrm{SM}} \circ (1 + (\mathbf{N} - \mathbf{D}).\mathbf{c}), & (6) \end{cases}
$$

Example

Here are examples of how to change the properties of the likelihood function with the likelihood type set to `variable Gaussian 0` in two different ways

```
a.attribute.update({'likelihood_type' : 0, 'devide' : False})
a.attribute.update({'likelihood_type' : 'variable Gaussian 0'})
```

1.2.3 Accessible variable

Here are the objects involved in data fitting that users can access after initializing the Fiteft object. While users don't need to pay attention to these values to use Fiteft, they provide convenience for observing the data of a specific experiment.

fiteft.C, fiteft.C2 : pandas Index
: Names of the parameters in the bases $\mathbf{c}$, $\mathbf{c}'$, respectively. This object is similar to a data `list`.

fiteft.obs : pandas DataFrame
: Table containing the experimental data.

fiteft.cor_exp : pandas DataFrame
: Table containing the correlation matrix data of the measured values provided by the experiment.

fiteft.cor_thep : pandas DataFrame
: Table containing the correlation matrix data of theoretical predictions, provided by the user. The default is the identity matrix.

fiteft.param : pandas DataFrame
: Table containing linear parameterization values, essentially all vectors $\mathbf{a}_i^{\mathrm{prod}\top}, \mathbf{a}_i^{\mathrm{decay}\top}, \mathbf{a}_i^{\mathrm{acc}\top}$ stacked together.

fiteft.rot : pandas DataFrame

Table containing the matrix $\overline{\mathbf{M}_{\lambda>a}}^{\top}$.

fiteft.Ndf,fiteft.Ddf,fiteft.Adf : pandas DataFrame

Tables containing the parameterization matrices $\mathbf{N}, \mathbf{D}, \mathbf{A}$ for basis c.

fiteft.Ndf2, fiteft.Ddf2, fiteft.Adf2 : pandas DataFrame

Tables containing the parameterization matrices $\mathbf{N}', \mathbf{D}', \mathbf{A}'$ for basis c'.

Example:
Below are some examples of how to use the mentioned data.
We can print the names of the coefficients tin the $\mathbf{c}'$ basis

```
>>> a.C2
Index(['c(3)Hq', 'c[1]HW-HB-HWB-HDD-uW-uB', 'c[2]HW-HB-HWB-HDD-uW-uB',
       'c[3]HW-HB-HWB-HDD-uW-uB', 'c[1]Hu-Hd-Hq(1)', 'c[1]Hl(1)-He',
       'c[1]Hl(3)-ll0', 'c[1]HG-uG-uH-top', 'c[2]HG-uG-uH-top',
       'c[3]HG-uG-uH-top'],
      dtype='object')
```

Afterwards, access the second parameter (note that parameter 0 is the first parameter)

```
>>> a.C2[2]
'c[2]HW-HB-HWB-HDD-uW-uB'
```

Access parameters 0 and 2 as follow

```
>>> a.C2[[0,2]]
Index(['c(3)Hq', 'c[2]HW-HB-HWB-HDD-uW-uB'], dtype='object')
```

To remove the second parameter and return the list of parameters:

```
>>> a.C2.drop(a.C2[2])
Index(['c(3)Hq', 'c[1]HW-HB-HWB-HDD-uW-uB', 'c[3]HW-HB-HWB-HDD-uW-uB',
       'c[1]Hu-Hd-Hq(1)', 'c[1]Hl(1)-He', 'c[1]Hl(3)-ll0', 'c[1]HG-uG-uH-top
 ↪',
       'c[2]HG-uG-uH-top', 'c[3]HG-uG-uH-top'],
      dtype='object')
```

To select columns 0 and 1, and rows 0 to 5 from the matrix $\mathbf{A}'$ using numerical indexing

```
>>> a.Adf2.iloc[0:5,[0,1]]
                                                                    c(3)Hq  c[1]HW-HB-HWB-HDD-uW-uB
production                       decay    acceptance signature
gg->H->Bgaga,0-jet,pTH<10GeV     H->gaga none        CS*BR/(CS_SM*BR_SM) -0.013             47.84019
```

```
gg->H->Bgaga,0-jet,10<=pTH<200GeV  H->gaga none       CS*BR/(CS_SM*BR_SM)  -0.013                  47.84019
gg->H->Bgaga,1-jet,pTH<60GeV       H->gaga none       CS*BR/(CS_SM*BR_SM)  -0.013                  47.84019
gg->H->Bgaga,1-jet,60<=pTH<120GeV  H->gaga none       CS*BR/(CS_SM*BR_SM)  -0.013                  47.84019
gg->H->Bgaga,1-jet,120<=pTH<200GeV H->gaga none       CS*BR/(CS_SM*BR_SM)  -0.013                  47.84019
```

To select columns corresponding to parameters 0 and 1, and rows 0 to 5 from the matrix $\mathbf{A}'$ using column names (characters), you can do the following

```
>>> temp = a.C2[[0,1]]
>>> temp
Index(['c(3)Hq', 'c[1]HW-HB-HWB-HDD-uW-uB'], dtype='object')
>>> a.Adf2.loc[:,temp].head()
                                                                       c(3)Hq  c[1]HW-HB-HWB-HDD-uW-uB
production                         decay   acceptance signature
gg->H->Bgaga,0-jet,pTH<10GeV       H->gaga none       CS*BR/(CS_SM*BR_SM)  -0.013                  47.84019
gg->H->Bgaga,0-jet,10<=pTH<200GeV  H->gaga none       CS*BR/(CS_SM*BR_SM)  -0.013                  47.84019
gg->H->Bgaga,1-jet,pTH<60GeV       H->gaga none       CS*BR/(CS_SM*BR_SM)  -0.013                  47.84019
gg->H->Bgaga,1-jet,60<=pTH<120GeV  H->gaga none       CS*BR/(CS_SM*BR_SM)  -0.013                  47.84019
gg->H->Bgaga,1-jet,120<=pTH<200GeV H->gaga none       CS*BR/(CS_SM*BR_SM)  -0.013                  47.84019
```

1.3 Computing function of Fiteft

1.3.1 fiteft.likelihood

```
fiteft.likelihood(C_df)
```

Argument:

C_df : Panda DataFrame
The table contains n vectors $\mathbf{c}'^\top$ stacked together, where the column names are parameter names. We specifically use columns that have names matching those in the object `fiteft.Ndf2`.

Return: **np.ndarray**
Returns an array of shape (n,1,1), where n is the number of vectors in `C_df`

Example:
To use this function, we need to understand how to initialize a pandas DataFrame `C_df` for the vector $\mathbf{c}'$. Below are two examples of how to initialize a pandas DataFrame:
Cách 1:

```
>>> df = pd.DataFrame(np.ones((2,2)), columns = ['c(3)Hq','c[1]Hl(3)-ll0'])
>>> df
   c(3)Hq  c[1]Hl(3)-ll0
0     1.0            1.0
1     1.0            1.0
```

Cách 2:

```
>>> df = pd.DataFrame(np.ones((2,2)), columns = a.C2[[0,6]])
```

```
>>> df
    c(3)Hq  c[1]Hl(3)-ll0
0      1.0            1.0
1      1.0            1.0
```

Afterwards, you can use the function `fiteft.likelihood` with the parameter as the pandas DataFrame defined as above

```
>>> a.likelihood(df)
array([[[1803.67453384]],

        [[1803.67453384]]])
```

The remaining parameters default to zero. The two numbers imply that the likelihood function is computed twice for the point $[\mathrm{c(3)Hq} = 1, \mathrm{c[1]Hl(3)} - \mathrm{ll0} = 1]$. We can assign a value of 0 to any parameter other than the initial two parameters, and the likelihood function will still return the same value.

```
>>> df = pd.DataFrame(np.array([[1,1,0],[1,1,0]]),\
    columns = ['c(3)Hq','c[1]Hl(3)-ll0','c[1]HW-HB-HWB-HDD-uW-uB'])
>>> df
    c(3)Hq  c[1]Hl(3)-ll0  c[1]HW-HB-HWB-HDD-uW-uB
0        1              1                        0
1        1              1                        0
>>> a.likelihood(df)
array([[[1803.67453384]],

        [[1803.67453384]]])
```

The order of the parameters can be arranged arbitrarily. The following are the results of $\mathcal{L}(\mathrm{c(3)Hq} = 0, \mathrm{c[1]Hl(3)} - \mathrm{ll0} = 1)$ and $\mathcal{L}(\mathrm{c(3)Hq} = 1, \mathrm{c[1]Hl(3)} - \mathrm{ll0} = 0)$ respectively

```
>>> df = pd.DataFrame(np.array([[0,1],[1,0]]),\
    columns = ['c(3)Hq','c[1]Hl(3)-ll0'])
>>> df
    c(3)Hq  c[1]Hl(3)-ll0
0        0              1
1        1              0
>>> a.likelihood(df)
array([[[  52.22424586]],

        [[1911.79581288]]])
```

Swapping the order of two parameters does not change the value of the likelihood function.

```
>>> df = pd.DataFrame(np.array([[1,0],[0,1]]),\
     columns = ['c[1]Hl(3)-1l0','c(3)Hq'])
>>> df
   c[1]Hl(3)-1l0  c(3)Hq
0              1       0
1              0       1
>>> a.likelihood(df)
array([[[  52.22424586]],

       [[1911.79581288]]])
```

1.3.2 fiteft.l

The purpose of this function is to make it easier to enter data without having to enter the names of the parameters each time the likelihood is calculated.

```
fiteft.l(cvecs)
```

Argument:

cvecs : pandas DataFrame or numpy ndarray
numpy ndarray: Of shape (n, m) representing n vectors $\mathbf{c}'^{\top}$ stacked column-wise, where m is the number of coefficients of the vector $\mathbf{c}'^{\top}$. The order of parameters in the numpy array will be automatically assigned to the columns of `cvecs`, following the order of parameters in `fiteft.Ndf2`.
pandas DataFrame: Similar to the argument of the function `fiteft.likelihood()`.

Return: **np.ndarray** with shape (n,1,1)
Array of shape (n, 1, 1), where n is the number of vectors in `cvecs`.

Example
To determine the number of parameters involved in computing the likelihood, you can access the columns of the parameterization matrix:

```
>>> a.C2
Index(['c(3)Hq', 'c[1]HW-HB-HWB-HDD-uW-uB', 'c[2]HW-HB-HWB-HDD-uW-uB',
       'c[3]HW-HB-HWB-HDD-uW-uB', 'c[1]Hu-Hd-Hq(1)', 'c[1]Hl(1)-He',
       'c[1]Hl(3)-1l0', 'c[1]HG-uG-uH-top', 'c[2]HG-uG-uH-top',
       'c[3]HG-uG-uH-top'],
      dtype='object')
```

The result returned is the parameters involved in fitting the data, corresponding to a total of 10 parameters. To determine the number of parameters, you can print the shape of the parameter matrix:

```
>>> a.C2.shape
(10,)
```

The returned result indicates that there are 10 parameters in $\mathbf{c}'$ is 10. From here, you can initialize a numpy ndarray of shape $(n, 10)$ as needed.

```
>>> arr = np.tile([[0],[0.1],[0.2]], (1,10))
>>> arr
array([[0. , 0. , 0. , 0. , 0. , 0. , 0. , 0. , 0. , 0. ],
       [0.1, 0.1, 0.1, 0.1, 0.1, 0.1, 0.1, 0.1, 0.1, 0.1],
       [0.2, 0.2, 0.2, 0.2, 0.2, 0.2, 0.2, 0.2, 0.2, 0.2]])
```

From here, we can use `arr` as argument for `fiteft.l()`

```
>>> a.l(arr)
array([[[   39.28929247]],

       [[ 6658.64606328]],

       [[25489.35871156]]])
```

The approach above will return a value equivalent to the following approach:

```
>>> arr = np.tile([[0],[0.1],[0.2]], (1,10))
>>> df = pd.DataFrame(arr, columns = a.C2)
>>> df
   c(3)Hq  c[1]HW-HB-HWB-HDD-uW-uB  ...  c[2]HG-uG-uH-top  c[3]HG-uG-uH-top
0     0.0                      0.0  ...               0.0               0.0
1     0.1                      0.1  ...               0.1               0.1
2     0.2                      0.2  ...               0.2               0.2

[3 rows x 10 columns]
>>> a.l(df)
array([[[   39.28929247]],

       [[ 6658.64606328]],

       [[25489.35871156]]])
```

1.3.3 fiteft.dl

This is a function for calculating the derivative $\mathcal{L}(\mathbf{c})$.

```
fiteft.dl(cvecs, delta =1.49e-08)
```

Argument:

cvecs : pandas DataFrame or numpy ndarray
numpy ndarray: Of shape (n, m) representing n vectors $\mathbf{c}'^{\top}$ stacked column-wise, where m is the number of coefficients of the vector $\mathbf{c}'^{\top}$. The order of parameters in the numpy array will be automatically assigned to the columns of `cvecs`, following the order of parameters in `fiteft.Ndf2`.
pandas DataFrame: the same as argument of the function `fiteft.likelihood()`.

Return: **np.ndarray** with the shape (n,1,m)
Array of shape (n, 1, m), where n is the number of vectors in `cvecs` and m is the number of parameters $\mathbf{c}'$. Returns a list of gradient vectors of the likelihood function with respect to the vectors $\mathbf{c}'$.

Example
We will compute the gradient of the likelihood function with respect to the vector $\mathbf{c}'$ with values set to 0. First, initialize `cvecs` representing the vectors $\mathbf{c}'$, then use `cvecs` as an argument for the `fiteft.dl` function

```
>>> arr = np.zeros((1,10))
>>> arr
array([[0., 0., 0., 0., 0., 0., 0., 0., 0., 0.]])
>>> a.dl(arr)
array([[[ 1.74947677e+01,  1.19368262e+03,  4.69186152e-01,
          4.96926121e+00, -5.59564136e+00,  2.77503991e+00,
         -1.56736581e+01,  1.22274234e+03, -8.44857583e+00,
         -8.51474262e+00]]])
>>> a.dl(arr).shape
(1, 1, 10)
```

The above numbers are equivalent to $\frac{\partial \mathcal{L}}{\partial \mathbf{c}'^{\top}}\big|_{c_i'=0}$
We also have another way to input data for the function as follows:

```
>>> df = pd.DataFrame([[0,0]], columns = ['c(3)Hq', 'c[1]HW-HB-HWB-HDD-uW-uB
↪'])
>>> df
   c(3)Hq  c[1]HW-HB-HWB-HDD-uW-uB
```

```
0        0                         0
>>> a.dl(df)
array([[[  17.49476769, 1193.68261566]]])
```

Another way with similar result

```
>>> df = pd.DataFrame([[0,0]], columns = a.C2[[0,1]])
>>> df
   c(3)Hq  c[1]HW-HB-HWB-HDD-uW-uB
0        0                         0
>>> a.dl(df)
array([[[  17.49476769, 1193.68261566]]])
```

Not listed values are defaulted to 0, we can switch the position of the vector in the dataframe to get the derivative values also

```
>>> df = pd.DataFrame([[0,0]], columns = ['c[1]HW-HB-HWB-HDD-uW-uB','c(3)Hq
 ↪'])
>>> df
   c[1]HW-HB-HWB-HDD-uW-uB  c(3)Hq
0                         0       0
>>> a.dl(df)
array([[[1193.68261566,   17.49476769]]])
```

We can also calculate the derivatives for many vectors at the same time

```
>>> df = pd.DataFrame([[0,0],[0.1,0.1]], columns = a.C2[[0,1]])
>>> df
   c(3)Hq  c[1]HW-HB-HWB-HDD-uW-uB
0     0.0                       0.0
1     0.1                       0.1
>>> a.dl(df)
array([[[1.74947686e+01, 1.19368262e+03]],

       [[6.14279192e+02, 5.29040790e+04]]])
```

1.3.4 fiteft.l_profile

```
fiteft.l_profile(cvecs, loc, val)
```

The function used to calculate the profile likelihood with $m-1$, with m is the number of parameters in the vector $\mathbf{c}'$. This function is equivalent to $\mathcal{L}(c'_i = \texttt{val}, \overline{\mathbf{c}'}_i)$, where $i = \texttt{loc}$.

Argument:

cvecs : pandas DataFrame or numpy ndarray

numpy ndarray: It has a shape of (n, m-1), representing n vectors $\overline{\mathbf{c}'}_i$ stacked column-wise, where $m-1$ is the number of elements of vector $\overline{\mathbf{c}'}_i^\top$. The order of parameters in the numpy array will automatically match those in `cvecs`, following the order in `fiteft.Ndf2` excluding ith coefficient.

pandas DataFrame: Similar to the argument of the function `fiteft.likelihood()`. Note that this DataFrame does not contain the parameter c'_i

loc : `str` or `int`

`str`: Name of the coefficient, such as `c(3)Hq`

`int`: Index of the parameter under consideration, starting from 0 as the first parameter.

val : `float`

Value of the parameter c'_i under consideration.

delta : `float`

Step size for computing the derivative, defaulting to 1.49e-8.

Return: **np.ndarray** with shape (n,1,1)
Array with shape (n,1,1), where n is number of vectors in `cvecs`

Example
We calculate $\mathcal{L}(c'_i = 0.5, \overline{\mathbf{c}'}_i)$, where $i =$`c(3)Hq` in two ways.
The first method involves using a dataframe

```
>>> df = pd.DataFrame([[0],[1]], columns =['c[1]HW-HB-HWB-HDD-uW-uB'])
>>> df
   c[1]HW-HB-HWB-HDD-uW-uB
0                        0
1                        1
>>> a.l_profile(df, 'c(3)Hq', 0.5)
array([[[   511.78960776]],

       [[261073.42566001]]])
```

The two values above represent $\mathcal{L}(\mathrm{c(3)Hq} = 0.5, \overline{\mathbf{c}'}_{\mathrm{c(3)Hq}})$, with `c[1]HW-HB-HWB-HDD-uW-uB` assigned to values 0 and 1, respectively, while other parameters default to 0. Using a DataFrame allows flexibility in arranging parameters.

Next, we will use a numpy array as input.

```
>>> arr = np.zeros((2,9))
>>> arr[1,0]=1
>>> arr
array([[0., 0., 0., 0., 0., 0., 0., 0., 0.],
       [1., 0., 0., 0., 0., 0., 0., 0., 0.]])
>>> a.l_profile(arr, loc = 0, val= 0.5)
array([[[   511.78960776]],

       [[261073.42566001]]])
```

or we can create a complete DataFrame simply as follows:

```
>>> arr = np.zeros((2,9))
>>> arr[1,0]=1
>>> df = pd.DataFrame(arr, columns = a.C2.drop(a.C2[0]))
>>> df
   c[1]HW-HB-HWB-HDD-uW-uB  c[2]HW-HB-HWB-HDD-uW-uB  ...  c[2]HG-uG-uH-top  c[3]HG-uG-uH-top
0                      0.0                      0.0  ...               0.0               0.0
1                      1.0                      0.0  ...               0.0               0.0

[2 rows x 9 columns]
>>> a.l_profile(df, a.C2[0], 0.5)
array([[[   511.78960776]],

       [[261073.42566001]]])
```

We obtain three identical results, where we set c'_i as the first value - position 0 - (corresponding to `c(3)Hq`) of the vector to be 0.5.

1.3.5 fiteft.dl_profile

```
fiteft.dl_profile(cvecs, loc, val, delta=1.49e-08)
```

Function for calculating derivative $\frac{\partial \mathcal{L}(c'_i=\texttt{val},\overline{\mathbf{c}'}_i)}{\partial}$, with $i = \texttt{loc}$

Argument:

cvecs : pandas DataFrame or numpy ndarray

numpy ndarray: Has shape $(n, m-1)$, representing n vectors $\overline{\mathbf{c}'}_i$ stacked column-wise, $m-1$ is the number of coefficent of $\overline{\mathbf{c}'}_i^\top$. The order of parameters in the numpy array will automatically match those in `cvecs`, following the order in `fiteft.Ndf2` excluding ith coefficient.

pandas DataFrame: Similar to the argument of the function `fiteft.likelihood()`. Note that this DataFrame does not contain the parameter c'_i.

loc : `str` or `int`
`str`: Name of the coefficient, such as `c(3)Hq`
`int`: Index of the parameter under consideration, starting from 0 as the first parameter.

val : `float`
Value of the parameter c'_i under consideration.

delta : `float`
Step size for computing the derivative, defaulting to 1.49e-8.

Return: **np.ndarray** with shape (n,1,m-1)
Array with shape (n,1,1), where n is number of vectors in `cvecs` and $m-1$ is the number of coefficients of $\overline{\mathbf{c}'_i}^\top$. Return a list of derivatives of the likelihood function with respect to $\overline{\mathbf{c}'_i}^\top$ vectors.

Example
Here, we calculate $\frac{\partial \mathcal{L}(c'_i=0.5,\overline{\mathbf{c}'}_i)}{\partial \overline{\mathbf{c}'}_i^\top}$, where $i =$`c(3)Hq`, in two ways.
First, we use dataframe as input

```
>>> df = pd.DataFrame([[0],[1]], columns =['c[1]HW-HB-HWB-HDD-uW-uB'])
>>> df
   c[1]HW-HB-HWB-HDD-uW-uB
0                        0
1                        1
>>> a.dl_profile(df, 'c(3)Hq', 0.5)
array([[[  2322.82750159]],

       [[518378.25472623]]])
```

The above two values represent $\frac{\partial \mathcal{L}(\hat{c'}_i,\overline{\mathbf{c}'}_i)}{\partial \overline{\mathbf{c}'}_i^\top}|_{\mathrm{c(3)Hq}=0.5}$, where the parameter `c[1]HW-HB-HWB-HDD-uW-uB` is assigned values 0 and 1, with other parameters defaulting to 0. Note that the function returns derivatives only for declared parameters. Using a DataFrame allows flexibility in parameter arrangement.

Next, we use numpy array as input

```
>>> arr = np.zeros((2,9))
>>> arr[1,0]=1
>>> arr
array([[0., 0., 0., 0., 0., 0., 0., 0., 0.],
```

```
       [1., 0., 0., 0., 0., 0., 0., 0., 0.]])
>>> a.dl_profile(arr, loc = 0, val= 0.5)
array([[[ 2.32282750e+03, -1.91485358e+02,  1.50639485e+01,
         -1.85706419e+02,  4.26760551e-01, -7.62017694e+01,
          7.84276547e+02, -3.75906673e-01,  1.47928321e+00]],

       [[ 5.18378259e+05,  2.66560225e+03,  4.41266808e+02,
         -6.12969642e+02, -1.40475791e+02, -4.58330052e+03,
          2.38905668e+05, -2.01389912e+03, -1.43328552e+03]]])
```

We have two visually different results, but if we focus on the first value of each row, we obtain the same result as before where we set c'_i to the first value (position 0) corresponding to `c(3)Hq` in the vector as 0 while other coefficients are defalted to 0.

Another approach is to use a complete dataframe containing two vectors $\overline{\mathbf{c}'}_i^\top$

```
>>> arr = np.zeros((2,9))
>>> arr[1,0]=1
>>> df = pd.DataFrame(arr, columns = a.C2.drop(a.C2[0]))
>>> df
   c[1]HW-HB-HWB-HDD-uW-uB  c[2]HW-HB-HWB-HDD-uW-uB  ...  c[2]HG-uG-uH-top  c[3]HG-uG-uH-top
0                      0.0                      0.0  ...               0.0               0.0
1                      1.0                      0.0  ...               0.0               0.0

[2 rows x 9 columns]
>>> a.dl_profile(df, a.C2[0], 0.5)
array([[[ 2.32282750e+03, -1.91485358e+02,  1.50639485e+01,
         -1.85706419e+02,  4.26760551e-01, -7.62017694e+01,
          7.84276547e+02, -3.75906673e-01,  1.47928321e+00]],

       [[ 5.18378255e+05,  2.66560225e+03,  4.41266808e+02,
         -6.12969642e+02, -1.40475791e+02, -4.58330052e+03,
          2.38905668e+05, -2.01389912e+03, -1.43328552e+03]]])
```

References

[1] "Combined measurements of Higgs boson production and decay using up to 139 fb^{-1} of proton-proton collision data at $\sqrt{s} = 13$ TeV collected with the ATLAS experiment, ATLAS-CONF-2120-053", (2021) (cit. on p. 3).

[2] "Interpretations of the combined measurement of Higgs boson production and decay, ATLAS-CONF-2020-053", (2020) (cit. on pp. 5–6).