

Fiteft manual

Contents

1	Data structure	2
1.1	observable.csv	3
1.2	correlation.csv	4
1.3	parametrization.csv	4
1.4	rotate.csv (optional)	6
2	The fiteft object	7
2.1	Initialize the fiteft	7
2.2	Changing attribute of the likelihood function	7
2.3	Accessible variable	8
3	Computing function of Fiteft	10
3.1	fiteft.likelihood	10
3.2	fiteft.l	12
3.3	fiteft.dl	14
3.4	fiteft.l_profile	15
3.5	fiteft.dl_profile	17

Fieft (**Fit** effective field theory) 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

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

- scipy

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[ATLAS:2021vrm].

The Fiteft database is stored in the directory

1 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.

corelation_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 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 " $\{\dots\}$ **decay**" where $\{\dots\}$ 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

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
$(\sigma_i \times B_Z)/B_Z^{SM}$	CS*BR_ZZ/BR_ZZSM	σ_i^{SM}
$B_i/B_{H \rightarrow ZZ}$	BR/BR_ZZ	$(B_i/B_Z)_{SM}$

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.

- +total_SM: SM theoretical prediction total error on the right side

1.2 correlation.csv

index	production decay	production decay	index ^T				
			gg->H->Bgaga,0-jet,10<=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	H->gaga	1.00	0.00	-0.02	-0.06	-0.02
gg->H->Bgaga,0-jet,10<=pTH<200GeV	H->gaga	H->gaga	0.00	1.00	-0.23	0.11	0.08
gg->H->Bgaga,1-jet,pTH<60GeV	H->gaga	H->gaga	-0.02	-0.23	1.00	0.24	0.14
gg->H->Bgaga,1-jet,60<=pTH<120GeV	H->gaga	H->gaga	-0.06	0.11	0.24	1.00	0.36
gg->H->Bgaga,1-jet,120<=pTH<200GeV	H->gaga	H->gaga	-0.02	0.08	0.14	0.36	1.00
gg->H->Bgaga,>=y2-jet,mjj<350GeV,pTH<120GeV	H->gaga	H->gaga	0.03	-0.04	-0.05	-0.19	-0.04
gg->H->Bgaga,>=y2-jet,mjj<350GeV,120<=pTH<200GeV	H->gaga	H->gaga	-0.02	0.08	-0.01	0.08	-0.02
gg->H->Bgaga,>=y2-jet,mjj>=y350GeV,pTH<200GeV	H->gaga	H->gaga	0.05	-0.02	0.00	-0.02	0.02
gg->H->Bgaga,200<=pTH<300GeV	H->gaga	H->gaga	-0.02	0.11	0.05	0.22	0.20
gg->H->Bgaga,300<=pTH<450GeV	H->gaga	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 [ATLAS:2020naq].

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 `observable.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.

1.3 parametrization.csv

Table 2 lists a subset of the parameters from the influence matrix **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

	Hbox	cHDD	cHG	c _u H	c _u G	c(3)Hl	c0l	cHWB	c(1)Hl	cHe	c(1)Hq	c(3)Hq	cHu	cHd	cHW
bin															
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,VHtopo	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

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 [ATLAS:2020naq] experiment. This table includes the impact of 34 SMEFT parameters.

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 $(\mathbf{a}_H^{\text{decay}})^\top$. 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:

$$\frac{\sigma^{\text{SMEFT}}(\text{gg} \rightarrow \text{H}, 0\text{jet}, p_{\text{TH}} < 10\text{GeV})|_{\text{cHbox}=1}}{\sigma^{\text{SM}}(\text{gg} \rightarrow \text{H}, 0\text{jet}, p_{\text{TH}} < 10\text{GeV})} = 1 + 0.12, \quad (1)$$

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

$$\begin{aligned} & \frac{\sigma^{\text{SMEFT}}(\text{gg} \rightarrow \text{H}, 0\text{jet}, p_H^{\text{T}} < 10\text{GeV})}{\sigma^{\text{SM}}(\text{gg} \rightarrow \text{H}, 0\text{jet}, p_H^{\text{T}} < 10\text{GeV})} \\ &= 1 + 0.12 \text{ cHbox} - 0.0294 \text{ cHDD} + 42.0 \text{ cHG} - 0.117 \text{ cuH} + 1.59 \text{ cuG} \\ &- 0.117 \text{ c(3)Hl} + 0.0587 \text{ c0ll} + 0.0 \text{ cHWB} + 0.0 \text{ c(1)Hl} + 0.0 \text{ cHe} + 0.0 \text{ c(1)Hq} \\ &+ 0.0 \text{ c(3)Hq} + 0.0 \text{ cHu} + 0.0 \text{ cHd} + 0.0 \text{ cHW} + 0.0 \text{ cHB} + 0.0 \text{ cG} \\ &+ 0.0 \text{ cuW} + 0.0 \text{ cuB} + 0.0 \text{ cqq} + 0.0 \text{ c(1)qq} + 0.0 \text{ c(3)qq} \\ &+ 0.0 \text{ c(31)qq} + 0.0 \text{ cuu} + 0.0 \text{ c(1)uu} + 0.0 \text{ c(1)ud} + 0.0 \text{ c(8)ud} + 0.0 \text{ c(1)qu} \\ &+ 0.0 \text{ c(8)qu} + 0.0 \text{ c(1)qd} + 0.0 \text{ c(8)qd} + 0.0 \text{ cW} + 0.0 \text{ cdH} + 0.0 \text{ ceH}. \end{aligned}$$

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.

2 The fiteft object

2.1 Initialize the fiteft

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

```
1 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

```
1 >>> from Fiteft import fiteft
2 >>> a = fiteft(experiment='ATLAS-CONF-2020-053')
3 Your input to the likelihood function is a DataFrame with one of these
  ↳ columns:
4 ['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)-l10',
  ↳ 'c[1]HG-uG-uH-top', 'c[2]HG-uG-uH-top', 'c[3]HG-uG-uH-top']
```

2.2 Changing attribute of the likelihood function

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

```
1 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'.

$$\Delta_y = \begin{cases} (\Delta_y^+ + \Delta_y^-)/2, & (2) \end{cases}$$

$$\Delta_y = \begin{cases} \sqrt{\frac{2\Delta^+ \circ \Delta^-}{\Delta^+ + \Delta^-} + \frac{2\Delta^+ - \Delta^-}{\Delta^+ + \Delta^-} \circ (y_c - y_{SM})}, & (3) \end{cases}$$

$$\Delta_y = \begin{cases} \sqrt{\Delta^+ \circ \Delta^- + (\Delta^+ - \Delta^-) \circ (y_c - y_{SM})}, & (4) \end{cases}$$

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).

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

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

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

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 **c**, **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 $\mathbf{a}_i^{\text{prod}\top}$, $\mathbf{a}_i^{\text{decay}\top}$, $\mathbf{a}_i^{\text{acc}\top}$ 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 in the c' basis

```
1 >>> a.C2
2 Index(['c(3)Hq', 'c[1]HW-HB-HWB-HDD-uW-uB', 'c[2]HW-HB-HWB-HDD-uW-uB',
3       'c[3]HW-HB-HWB-HDD-uW-uB', 'c[1]Hu-Hd-Hq(1)', 'c[1]H1(1)-He',
4       'c[1]H1(3)-l10', 'c[1]HG-uG-uH-top', 'c[2]HG-uG-uH-top',
5       'c[3]HG-uG-uH-top'],
6       dtype='object')
```

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

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

Access parameters 0 and 2 as follow

```
1 >>> a.C2[[0,2]]
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:

```
1 >>> a.C2.drop(a.C2[2])
2 Index(['c(3)Hq', 'c[1]HW-HB-HWB-HDD-uW-uB', 'c[3]HW-HB-HWB-HDD-uW-uB',
3       'c[1]Hu-Hd-Hq(1)', 'c[1]H1(1)-He', 'c[1]H1(3)-l10', 'c[1]HG-uG-uH-top
4       ↪',
5       'c[2]HG-uG-uH-top', 'c[3]HG-uG-uH-top'],
6       dtype='object')
```

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

```
1 >>> a.Adf2.iloc[0:5,[0,1]]
2
3 production          decay  acceptance  signature      c(3)Hq  c[1]HW-HB-HWB-HDD-uW-uB
4 gg->H->Bgaga,0-jet,pTH<10GeV  H->gaga none      CS*BR/(CS_SM*BR_SM)  -0.013      47.84019
```

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

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

```

1 >>> temp = a.C2[[0,1]]
2 >>> temp
3 Index(['c(3)Hq', 'c[1]HW-HB-HWB-HDD-uW-uB'], dtype='object')
4 >>> a.Adf2.loc[:,temp].head()
5
6 production      decay      acceptance      signature      c(3)Hq      c[1]HW-HB-HWB-HDD-uW-uB
7 gg->H->Bgaga,0-jet,pTH<10GeV      H->gaga none      CS*BR/(CS_SM*BR_SM)      -0.013      47.84019
8 gg->H->Bgaga,0-jet,10<=pTH<200GeV      H->gaga none      CS*BR/(CS_SM*BR_SM)      -0.013      47.84019
9 gg->H->Bgaga,1-jet,pTH<60GeV      H->gaga none      CS*BR/(CS_SM*BR_SM)      -0.013      47.84019
10 gg->H->Bgaga,1-jet,60<=pTH<120GeV      H->gaga none      CS*BR/(CS_SM*BR_SM)      -0.013      47.84019
11 gg->H->Bgaga,1-jet,120<=pTH<200GeV      H->gaga none      CS*BR/(CS_SM*BR_SM)      -0.013      47.84019

```

3 Computing function of Fiteft

3.1 fiteft.likelihood

```

1 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:

```

1 >>> df = pd.DataFrame(np.ones((2,2)), columns = ['c(3)Hq', 'c[1]H1(3)-110'])
2 >>> df
3      c(3)Hq      c[1]H1(3)-110
4 0         1.0         1.0
5 1         1.0         1.0

```

Cách 2:

```

1 >>> df = pd.DataFrame(np.ones((2,2)), columns = a.C2[[0,6]])
2 >>> df
3      c(3)Hq  c[1]Hl(3)-ll0
4 0         1.0             1.0
5 1         1.0             1.0

```

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

```

1 >>> a.likelihood(df)
2 array([[1803.67453384]],
3
4       [[1803.67453384]])

```

The remaining parameters default to zero. The two numbers imply that the likelihood function is computed twice for the point $[c(3)Hq = 1, c[1]Hl(3) - ll0 = 1]$. 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.

```

1 >>> df = pd.DataFrame(np.array([[1,1,0],[1,1,0]]),\
2      columns = ['c(3)Hq', 'c[1]Hl(3)-ll0', 'c[1]HW-HB-HWB-HDD-uW-uB'])
3 >>> df
4      c(3)Hq  c[1]Hl(3)-ll0  c[1]HW-HB-HWB-HDD-uW-uB
5 0         1             1             0
6 1         1             1             0
7 >>> a.likelihood(df)
8 array([[1803.67453384]],
9
10      [[1803.67453384]])

```

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

```

1 >>> df = pd.DataFrame(np.array([[0,1],[1,0]]),\
2      columns = ['c(3)Hq', 'c[1]Hl(3)-ll0'])
3 >>> df
4      c(3)Hq  c[1]Hl(3)-ll0
5 0         0             1
6 1         1             0
7 >>> a.likelihood(df)
8 array([[ 52.22424586]],
9

```

```
10 [[1911.79581288]]])
```

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

```
1 >>> df = pd.DataFrame(np.array([[1,0],[0,1]]),\
2     columns = ['c[1]Hl(3)-110', 'c(3)Hq'])
3 >>> df
4     c[1]Hl(3)-110  c(3)Hq
5 0                1      0
6 1                0      1
7 >>> a.likelihood(df)
8 array([[ 52.22424586]],
9
10 [[1911.79581288]]])
```

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.

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

Argument:

cvecs : pandas DataFrame or numpy ndarray

numpy ndarray: Of shape (n, m) representing n vectors \mathbf{c}'^T stacked column-wise, where m is the number of coefficients of the vector \mathbf{c}'^T . 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:

```
1 >>> a.C2
2 Index(['c(3)Hq', 'c[1]HW-HB-HWB-HDD-uW-uB', 'c[2]HW-HB-HWB-HDD-uW-uB',
3       'c[3]HW-HB-HWB-HDD-uW-uB', 'c[1]Hu-Hd-Hq(1)', 'c[1]Hl(1)-He',
```

```

4      'c[1]H1(3)-110', 'c[1]HG-uG-uH-top', 'c[2]HG-uG-uH-top',
5      'c[3]HG-uG-uH-top'],
6      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:

```

1 >>> a.C2.shape
2 (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.

```

1 >>> arr = np.tile([[0],[0.1],[0.2]], (1,10))
2 >>> arr
3 array([[0. , 0. , 0. , 0. , 0. , 0. , 0. , 0. , 0. , 0. ],
4        [0.1, 0.1, 0.1, 0.1, 0.1, 0.1, 0.1, 0.1, 0.1, 0.1],
5        [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()`

```

1 >>> a.l(arr)
2 array([[ 39.28929247],
3
4        [ 6658.64606328]],
5
6        [[25489.35871156]])

```

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

```

1 >>> arr = np.tile([[0],[0.1],[0.2]], (1,10))
2 >>> df = pd.DataFrame(arr, columns = a.C2)
3 >>> df
4      c(3)Hq  c[1]HW-HB-HWB-HDD-uW-uB  ...  c[2]HG-uG-uH-top  c[3]HG-uG-uH-top
5 0      0.0      0.0  ...      0.0      0.0
6 1      0.1      0.1  ...      0.1      0.1
7 2      0.2      0.2  ...      0.2      0.2
8
9 [3 rows x 10 columns]
10 >>> a.l(df)
11 array([[ 39.28929247],
12
13        [ 6658.64606328]],
14
15        [[25489.35871156]])

```

3.3 fiteft.dl

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

```
1 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

```
1 >>> arr = np.zeros((1,10))
2 >>> arr
3 array([[0., 0., 0., 0., 0., 0., 0., 0., 0., 0.]])
4 >>> a.dl(arr)
5 array([[[ 1.74947677e+01,  1.19368262e+03,  4.69186152e-01,
6           4.96926121e+00, -5.59564136e+00,  2.77503991e+00,
7           -1.56736581e+01,  1.22274234e+03, -8.44857583e+00,
8           -8.51474262e+00]]]])
9 >>> a.dl(arr).shape
10 (1, 1, 10)
```

The above numbers are equivalent to $\frac{\partial \mathcal{L}}{\partial \mathbf{c}'^\top} |_{\mathbf{c}'=0}$

We also have another way to input data for the function as follows:

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

```

4 0      0      0
5 >>> a.dl(df)
6 array([[ 17.49476769, 1193.68261566]])

```

Another way with similar result

```

1 >>> df = pd.DataFrame([[0,0]], columns = a.C2[[0,1]])
2 >>> df
3      c(3)Hq   c[1]HW-HB-HWB-HDD-uW-uB
4 0      0      0
5 >>> a.dl(df)
6 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

```

1 >>> df = pd.DataFrame([[0,0]], columns = ['c[1]HW-HB-HWB-HDD-uW-uB', 'c(3)Hq
      ↪'])
2 >>> df
3      c[1]HW-HB-HWB-HDD-uW-uB   c(3)Hq
4 0      0      0
5 >>> a.dl(df)
6 array([[1193.68261566, 17.49476769]])

```

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

```

1 >>> df = pd.DataFrame([[0,0],[0.1,0.1]], columns = a.C2[[0,1]])
2 >>> df
3      c(3)Hq   c[1]HW-HB-HWB-HDD-uW-uB
4 0      0.0      0.0
5 1      0.1      0.1
6 >>> a.dl(df)
7 array([[1.74947686e+01, 1.19368262e+03],
8
9        [[6.14279192e+02, 5.29040790e+04]])

```

3.4 fiteft.l_profile

```

1 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 $\mathcal{L}(c'_i = \text{val}, \overline{\mathbf{c}}'_i)$, where $i = \text{loc}$.

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 $\overline{\mathbf{c}}'_i$. The order of parameters in the numpy array will automatically match those in **cvecs**, following the order in `fiteft.Ndf2` excluding i th 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 $1.49\text{e-}8$.

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 $i = \text{c(3)Hq}$ in two ways.

The first method involves using a dataframe

```

1 >>> df = pd.DataFrame([[0],[1]], columns=['c[1]HW-HB-HWB-HDD-uW-uB'])
2 >>> df
3      c[1]HW-HB-HWB-HDD-uW-uB
4 0                                0
5 1                                1
6 >>> a.l_profile(df, 'c(3)Hq', 0.5)
7 array([[ 511.78960776],
8
9        [[261073.42566001]]])

```

The two values above represent $\mathcal{L}(c(3)Hq = 0.5, \overline{\mathbf{c}}'_{c(3)Hq})$, 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.

```

1 >>> arr = np.zeros((2,9))
2 >>> arr[1,0]=1
3 >>> arr
4 array([[0., 0., 0., 0., 0., 0., 0., 0., 0.],
5        [1., 0., 0., 0., 0., 0., 0., 0., 0.]])
6 >>> a.l_profile(arr, loc = 0, val= 0.5)
7 array([[ 511.78960776]],
8
9        [[261073.42566001]]])

```

or we can create a complete DataFrame simply as follows:

```

1 >>> arr = np.zeros((2,9))
2 >>> arr[1,0]=1
3 >>> df = pd.DataFrame(arr, columns = a.C2.drop(a.C2[0]))
4 >>> df
5      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-
6 0                                0.0                                0.0  ...                0.0                0.0
7 1                                1.0                                0.0  ...                0.0                0.0
8
9 [2 rows x 9 columns]
10 >>> a.l_profile(df, a.C2[0], 0.5)
11 array([[ 511.78960776]],
12
13        [[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.

3.5 fiteft.dl_profile

```

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

```

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

Argument:

cvecs : pandas DataFrame or numpy ndarray

numpy ndarray: Has shape $(n, m - 1)$, representing n vectors \bar{c}_i stacked column-wise, $m - 1$ is the number of coefficient of \bar{c}_i^\top . The order of parameters in the numpy array will automatically match those in **cvecs**, following the order

in `fiteft.Ndf2` excluding i th 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 $i = \text{c}(3)\text{Hq}$, in two ways.

First, we use `dataframe` as input

```
1 >>> df = pd.DataFrame([[0],[1]], columns = ['c[1]HW-HB-HWB-HDD-uW-uB'])
2 >>> df
3      c[1]HW-HB-HWB-HDD-uW-uB
4 0                               0
5 1                               1
6 >>> a.dl_profile(df, 'c(3)Hq', 0.5)
7 array([[ [ 2322.82750159]],
8
9         [[518378.25472623]]])
```

The above two values represent $\frac{\partial \mathcal{L}(\hat{c}'_i, \overline{\mathbf{c}}'_i)}{\partial \overline{\mathbf{c}}'_i{}^\top} \big|_{\text{c}(3)\text{Hq}=0.5}$, 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

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

```

3 >>> arr
4 array([[0., 0., 0., 0., 0., 0., 0., 0., 0.],
5        [1., 0., 0., 0., 0., 0., 0., 0., 0.]])
6 >>> a.dl_profile(arr, loc = 0, val= 0.5)
7 array([[[ 2.32282750e+03, -1.91485358e+02,  1.50639485e+01,
8           -1.85706419e+02,  4.26760551e-01, -7.62017694e+01,
9           7.84276547e+02, -3.75906673e-01,  1.47928321e+00]],
10
11         [[ 5.18378259e+05,  2.66560225e+03,  4.41266808e+02,
12           -6.12969642e+02, -1.40475791e+02, -4.58330052e+03,
13           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 $c(3)H_q$ in the vector as 0 while other coefficients are defaulted to 0.

Another approach is to use a complete dataframe containing two vectors \overline{c}_i^T

```

1 >>> arr = np.zeros((2,9))
2 >>> arr[1,0]=1
3 >>> df = pd.DataFrame(arr, columns = a.C2.drop(a.C2[0]))
4 >>> df
5      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-
6 0                                0.0                                0.0  ...                0.0                0.0
7 1                                1.0                                0.0  ...                0.0                0.0
8
9 [2 rows x 9 columns]
10 >>> a.dl_profile(df, a.C2[0], 0.5)
11 array([[[ 2.32282750e+03, -1.91485358e+02,  1.50639485e+01,
12           -1.85706419e+02,  4.26760551e-01, -7.62017694e+01,
13           7.84276547e+02, -3.75906673e-01,  1.47928321e+00]],
14
15         [[ 5.18378255e+05,  2.66560225e+03,  4.41266808e+02,
16           -6.12969642e+02, -1.40475791e+02, -4.58330052e+03,
17           2.38905668e+05, -2.01389912e+03, -1.43328552e+03]]])

```
