FeHiPro 1.0.0

Stefan Bucherer

December 16, 2009

Abstract

This is the manual of the FORTRAN code FeHiPro 1.0.0, a code for computing inclusive cross-section and any differentiable distribution for Higgs production in gluon fusion including decay into $\gamma\gamma$, $WW \to \ell\nu\ell\nu$ and $ZZ \to \ell\ell\ell\ell$.

1 Introduction

The FeHiPro package consists of the following gzipped tar files:

- FeHiPro.tgz: Contains the source files for the program.
- pdfdat.tgz: Contains grid files for PDF sets supported by FeHiPro.

Download these files from www.phys.ethz.ch/~pheno/fehipro and unpack them in your directory structure. The relative paths are irrelevant. Now you should have created the following directories:

- /path1/FeHiPro
- /path2/pdfdat

For installing the package, you need to follow the following steps:

1. **Install Cuba library.**

   ```
   > cd /path1/FeHiPro/Cuba-1.6_modified
   > ./configure
   > make
   ```

   For details concerning the Cuba library, refer to [1].
   Note: Ignore the following (or similar) error message:

   ```
   ...
   strip: 'Vegas': No such file
   make: *** [Vegas] Error 1
   ```

2. **Install FeHiPro.**

   ```
   > cd /path1/FeHiPro
   ```

   Then open the makefile and choose your Fortran and C/C++ compiler in the user input section. Supported are g77, gfortran and ifort as Fortran compilers and gcc, g++ and icc as C/C++ compilers.
   To compile the code, type

   ```
   > make
   ```

This can take several minutes. Four executables get created in `FeHiPro/bin`: `fehip` (using analysis routines in `analysisSrc`), `fehipgg`(using analysis routines in `analysisSrcgg`), `fehipWW`(using analysis routines in `analysisSrcWW`) and `fehipZZ`(using analysis routines in `analysisSrcZZ`). The object files are found in `FeHiPro/build`, `FeHiPro/mainBuild` and `FeHiPro/pvegas/build`.

If you have modified the analysis routines in the `analysisSrcXX` subdirectories, it is recommended to remove the `mainBuild` subdirectory and then to recompile the code:

```
> make clean
> make
```

If you want to remove all object files, type

```
> make wipe
```

2 Running the code

The computation in the $m_t = \infty$ limit is performed using sector decomposition. For better convergence, these sectors are integrated separately as inpendent jobs. This leads to 96 sectors for the computation in the limit of an effective theory. The mass corrections are computed through defining additional 3 sectors. However, these sectors are not related to sector decomposition.

In order to easily organize the computation we provide Python scripts in the `scripts` subdirectory. We strongly suggest to use these scripts. By default, four scripts for submitting jobs are provided, `DoRunTot.py`, `DoRungg.py`, `DoRunWW.py` and `DoRunZZ.py` for the computation of the total cross-section using the fast inclusive option and exclusive computations in the di-photon decay channel, the WW channel as well as the ZZ channel, respectively. These scripts depend on class definitions in `DoRunClasses.py`. The idea is that the user does not have to touch the implementation in `DoRunClasses.py` but modifying the simple run scripts `DoRunXX.py` suffices for running any computation.

After modification of the run script a computation is started by

```
>chmod u+x DoRunXX.py
>./DoRunXX.py
```

In the following subsection we are going to describe the various options.

Please be aware that the vegas integrators write out state files allowing to resume an integration when interrupted. Therefore make sure to erase these state files if you want to start a job with different parameters in the same directory.

2.1 Options

Job steering.

- `jobdefs['projdir']`: Path to the directory in which to run the jobs and to contain the results. If it does not exist, yet, it gets created.
- `jobdefs['project']`: Subdirectory of `jobdefs['projdir']` for this particular job.
- `jobdefs['exepath']`: Path to the directory containing the `FeHiPro` executable (the `'bin'` subdirectory).
- `jobdefs['gridpath']`: Path to the directory containing the PDF grid files.
- `jobdefs['exe']`: Name of the executable.
- `jobdefs['ifcluster']`: Boolean flag. If set to `True`, the script assumes the job to be executed on a cluster using `bsub`. You might want to check the implementation of class `DoRun` and its method `RunCluster` in `DoRunClasses.py`. If set to `False`, all sub-jobs will be executed sequentially.

- `jobdefs['walltime']`: Estimated run-time in minutes. This value is passed as the `-W` option to `bsub`. Has no effect if `jobdefs['ifcluster']=False`.
- `jobdefs['submitts']`: Number of submissions of this job. Used for splitting up long jobs into several shorter ones. Each submission waits for the previous one to be finished. Has no effect if `jobdefs['ifcluster']=False`.

Defining a scan. A scan in any parameter described in the next paragraph is defined via the array `myscan`. Its first entry must contain the parameter name, while the remaining entries contain the values for the scan.

If there are parameters that depend on the value of the scan variable, these can be defined with the Python dictionnary `myvars` of the form

```
myvars = {'scanval1': {'param1': val1a, 'param2': val2a, ...},
          'scanval2': {'param1': val1b, 'param2': val2b, ...}, ...}
```

Consider for example the scan in the renormalization and factorization scale $\mu_{\rm R} = \mu_{\rm F} = \mu$, $m_{\rm h}/2 \leq \mu \leq 2m_{\rm h}$. In this case we would define `myscan` and `myvars` as follows:

```
myscan = ['muRrel','0.5','1.0','2.0']
myvars = {'0.5': {'muFrel': '0.5'}, '1.0': {'muFrel': '1.0'},  '2.0': {'muFrel': '2.0'}}
```

Note that parameter values defined in `myvars` and `myscan` overwrite values defined in the list described below.

Parameters. Listed are also the default values.

- `myfixed['porder']= '2'`:
  Order of the calculation. 0: LO; 1: NLO; 2: NNLO.
- `myfixed['iinclusive']= '1'`:
  Flag for choosing the fast inclusive option for NNLO computation (0: exclusive computation; 1: fast inclusive computation).
- `myfixed['mh']= '120.0'`:
  Higgs mass in GeV.
- `myfixed['GammaH']= '0'`:
  Width of the Higgs in GeV. If `myfixed['GammaH']= '0'`, HDECAY is used to compute the width. Note, that decays are computed in the narrow width approximation.
- `myfixed['muFrel']= '0.5'`:
  Relative factorization scale $\mu_{\rm F}^{\rm rel}$, $\mu_{\rm F} = \mu_{\rm F}^{\rm rel} m_{\rm h}$.
- `myfixed['muRrel']= '0.5'`:
  Relative renormalization scale $\mu_{\rm R}^{\rm rel}$, $\mu_{\rm R} = \mu_{\rm R}^{\rm rel} m_{\rm h}$.
- `myfixed['icollider']= '0'`:
  pp (0) or $p\bar{p}$ (1) collider.
- `myfixed['sqrtS']= '14000'`:
  Center of mass energy of colliding hadrons in GeV.
- `myfixed['mW']= '80.398'`:
  Mass of the W boson in GeV.
- `myfixed['GammaW']= '2.141'`:
  Width of the W boson in GeV.
- `myfixed['mZ']= '91.1876'`:
  Mass of the Z boson in GeV
- `myfixed['GammaZ']= '2.4952'`:
  Width of the Z boson in GeV.

- `myfixed['mt']= '173.1'`:
  Mass of the top quark in GeV.
- `myfixed['imbscheme']= '0'`:
  Renormalization scheme for bottom quark.
- `myfixed['mb']= '4.8'`:
  If `myfixed['imbscheme']= '0'`: Pole mass of the bottom quark in GeV. If `myfixed['imbscheme']= '1'`: $\overline{\text{MS}}$ mass of the bottom quark at 10 GeV in GeV.
- `myfixed['imassive']= '1'`:
  Flag for computing mass corrections through NLO (0: no, 1: yes).
- `myfixed['iewk']= '2'`:
  Flag for computing electroweak contributions (0: no, 1: only for virtual contributions, 2: for virtual and real contributions).
- `myfixed['ytrel']= '1.0'`:
  Yukawa coupling between Higgs boson and top quarks normalized to the Standard Model.
- `myfixed['ybrel']= '1.0'`:
  Yukawa coupling between Higgs boson and bottom quarks normalized to the Standard Model.
- `myfixed['idecay']= '0'`:
  Flag for choice of the decay. 0: no decay; 1: $h \to WW \to \ell\nu\ell\nu$; 2: $h \to WW/ZZ \to \ell\nu\ell\nu$; 3: $h \to ZZ \to \ell\ell\ell\ell$; 4: $h \to ZZ \to \ell\ell\ell'\ell'$; 5: $h \to \gamma\gamma$.
- `myfixed['exclWidth']= '0'`:
  Flag for excluding width of the electroweak gauge bosons in (0: integrate over width of electroweak gauge bosons; 1: narrow width approximation for electroweak gauge bosons).
- `myfixed['BRgg']= '0.0'`:
  Branching ratio for $h \to \gamma\gamma$. If `myfixed['BRgg']= '0'`, HDECAY is used to compute the branching ratio.
- `myfixed['intMethod']= '11'`:
  Choice of the integrator. 1: `nvegas`, 11: modified Cuba vegas.
- `myfixed['accRel']= '0.1'`:
  Accepted relative error in %.
- `myfixed['accAbs']= '0.000001'`:
  Accepted absolute error.
- `myfixed['ntherm']= '10'`:
  Number of thermalization iterations.
- `myfixed['nrefin']= '5'`:
  Number of refinement iterations (only for `nvegas`).
- `myfixed['ncalls']= '50000'`:
  Number of integrand evaluation per iteration.
- `myfixed['nincr']= '0'`:
  Increase of number of calls per iteration.
- `myfixed['adaptofirst']= '1'`:
  Flag to adapt only to the first component (1) or to phase space discriminant (0).
- `myfixed['maxtime']= '0'`:
  Maximal time for integration (only for `nvegas`).
- `myfixed['pdfset']= '4.0'`:
  Choice of PDF set. 4.0= MSTW 2008; 3.0= MRST 2006; 2.0= MRST 2004; 1.0= MRST 2001.

- `myfixed['pdferror']= '0'`:
  Flag for computing PDF error (only for MRST 2006 and MSTW 2008). 0: No determination of PDF error; 1: Computation of PDF error.
- `myfixed['whichalphas']= '0'`:
  Choice of α_S value (depending on choice of PDF set). Only for MSTW 2008 sets: 0: central value, 1: α_S at -(95% C.L.) level; 2: α_S at half the -(95% C.L.) level; 3: α_S at half the +(95% C.L.) level; 4: α_S at the +(95% C.L.) level;
- `myfixed['C0user']= '0.0'`, `myfixed['C1user']= '0.0'` and `myfixed['C2user']= '0.0'`:
  Customize the Wilson coefficient for the computation in the effective theory, where the complete Wilson coefficient is given by

$$C_W = C_0^{\rm SM} + C_0^{\rm user} + \frac{\alpha_s}{\pi}\left(C_1^{\rm SM} + C_1^{\rm user}\right) + \left(\frac{\alpha_s}{\pi}\right)^2\left(C_2^{\rm SM} + C_2^{\rm user}\right) . \qquad (1)$$

  The Standard Model coefficients are normalized such that:

$$C_0^{\rm SM} = 1 + \delta_{\rm EW}$$
$$C_1^{\rm SM} = \frac{11}{4} + \delta_{\rm EW}\frac{7}{6}$$
$$C_2^{\rm SM} = \frac{2777}{288} + \frac{19}{16}\left(\ln\frac{\mu_R^2}{m_{\rm h}^2} + \ln\frac{m_{\rm h}^2}{m_{\rm t}^2}\right) + 5\left(-\frac{67}{96} + \frac{1}{3}\left(\ln\frac{\mu_R^2}{m_{\rm h}^2} + \ln\frac{m_{\rm h}^2}{m_{\rm t}^2}\right)\right) + \delta_{\rm EW}(?)\,.$$

  The terms proportional to $\delta_{\rm EW}$ are due to graphs involving electroweak gauge bosons coupling to the Higgs and are computed if `myfixed['iewk']='1.0'`.

Cut parameters.

- `myfixed['PtHiggsCut']= '0.0'`:
  Cut on the Higgs transverse momentum. 0: no cut; 1: apply cut.
  `myfixed['PtHiggsMin']= '0.0'`: Lower value for $p_{\rm T}^H$ in GeV. Has no effect if `myfixed['PtHiggsCut']= '0.0'`.
  `myfixed['PtHiggsMax']= '0.0'`: Upper value for $p_{\rm T}^H$ in GeV. Has no effect if `myfixed['PtHiggsCut']= '0.0'`.
- `myfixed['EtaHiggsCut']= '0.0'`:
  Cut on the Higgs rapidity. 0: no cut; 1: apply cut.
  `myfixed['EtaHiggsMin']= '0.0'`: Lower value for y^H. Has no effect if `myfixed['EtaHiggsCut']= '0.0'`.
  `myfixed['EtaHiggsMax']= '0.0'`: Upper value for y^H. Has no effect if `myfixed['EtaHiggsCut']= '0.0'`.
- `myfixed['JetConeSize']= '0.0'`:
  Size of the jet cone.
- `myfixed['JetDefPtMin']= '0.0'`:
  Minimum $p_{\rm T}$ for a jet in GeV.
- `myfixed['JetDefEtaMax']='0.0'`:
  Maximum pseudorapidity for a jet.
- `myfixed['UseSISCone']= '0.0'`:
  Flag to choose the cone algorithm. 0: Simple cone algorithm; 1: SISCone algorithm.
- `myfixed['JetVeto']= '0.0'`:
  Apply a jet veto. 0: no veto; 1: apply veto.
  `myfixed['JetPtMax']= '0.0'`: Maximal allowed transverse momentum of jet for jet veto in GeV.
  `myfixed['JetEtaMax']= '0.0'`: Maximal allowed pseudorapidity of jet for jet veto.
- `myfixed['NumVeto']= '0.0'`:
  Cut on the number of jets. 0: No cut; 1: apply cut.
  `myfixed['JetNmax']= '0'`: Maximal number of jets.

- `myfixed['TeVVeto']= '0.0'`:
  Flag for special Tevatron jet veto. (TO BE EXPLAINED.) 0: no cut; 1: apply cut.
  `myfixed['TeVJetPtMax1']= '0.0'`: Maximum transverse momentum for jet 1 in GeV for special Tevatron jet veto.
  `myfixed['TeVJetPtMax2']= '0.0'`: Maximum transverse momentum for jet 2 in GeV for special Tevatron jet veto.
- `myfixed['MassV1Cut']= '0.0'`:
  Cut on the mass of first electroweak gauge boson. 0: no cut; 1: apply cut.
  `myfixed['MassV1Min']= '0.0'`: Minimum mass of first electroweak gauge boson in GeV.
  `myfixed['MassV1Max']= '0.0'`: Maximum mass of first electroweak gauge boson in GeV.
- `myfixed['MassV2Cut']= '0.0'`:
  Cut on the mass of second electroweak gauge boson. 0: no cut; 1: apply cut.
  `myfixed['MassV2Min']= '0.0'`: Minimum mass of second electroweak gauge boson in GeV.
  `myfixed['MassV2Max']= '0.0'`: Maximum mass of second electroweak gauge boson in GeV.
- `myfixed['PtV1Cut']= '0.0'`:
  Cut on transverse momentum of first electroweak gauge boson. 0: no cut; 1: apply cut.
  `myfixed['PtV1Min']= '0.0'`: Minimum transverse momentum of first electroweak gauge boson in GeV.
  `myfixed['PtV1Max']= '0.0'`: Maximum transverse momentum of first electroweak gauge boson in GeV.
- `myfixed['PtV2Cut']= '0.0'`:
  Cut on transverse momentum of second electroweak gauge boson. 0: no cut; 1: apply cut.
  `myfixed['PtV2Min']= '0.0'`: Minimum transverse momentum of second electroweak gauge boson in GeV.
  `myfixed['PtV2Max']= '0.0'`: Maximum transverse momentum of second electroweak gauge boson in GeV.
- `myfixed['EtaV1Cut']= '0.0'`:
  Cut on rapidity of first electroweak gauge boson. 0: no cut; 1: apply cut.
  `myfixed['EtaV1Min']= '0.0'`: Minimum rapidity of first electroweak gauge boson.
  `myfixed['EtaV1Max']= '0.0'`: Maximum rapidity of first electroweak gauge boson.
- `myfixed['EtaV2Cut']= '0.0'`:
  Cut on rapidity of second electroweak gauge boson. 0: no cut; 1: apply cut.
  `myfixed['EtaV2Min']= '0.0'`: Minimum rapidity of second electroweak gauge boson.
  `myfixed['EtaV2Max']= '0.0'`: Maximum rapidity of second electroweak gauge boson.
- `myfixed['LepHardMinPtCut']= '0.0'`:
  Cut on minimal transverse momentum of the harder lepton. 0: no cut; 1: apply cut.
  `myfixed['LepHardPtMinVal']= '0.0'`: Minimal transverse momentum of harder lepton in GeV.
- `myfixed['LepHardMaxPtCut']= '0.0'`:
  Cut on maximal transverse momentum of harder lepton. 0: no cut; 1: apply cut.
  `myfixed['LepHardPtMaxVal']= '0.0'`: Maximal transverse momentum of harder lepton in GeV.
- `myfixed['LepSoftMinPtCut']= '0.0'`:
  Cut on minimal transverse momentum of softer lepton. 0: no cut; 1: apply cut.
  `myfixed['LepSoftPtMinVal']= '0.0'`: Minimal transverse momentum of softer lepton in GeV.
- `myfixed['LepSoftMaxPtCut']= '0.0'`:
  Cut on maximal transverse momentum of softer lepton. 0: no cut; 1: apply cut.

`myfixed['LepSoftPtMaxVal']= '0.0'`: Maximal transverse momentum of softer lepton in GeV.

- `myfixed['LepHardMaxEtaCut']= '0.0'`:
  Cut on maximal pseudorapidity of harder lepton. 0: no cut; 1: apply cut.
  `myfixed['LepHardEtaMaxVal']= '0.0'`: Maximal pseudorapidity of harder lepton.
- `myfixed['LepSoftMaxEtaCut']= '0.0'`:
  Cut on maximal pseudorapidity of softer lepton. 0: no cut; 1: apply cut.
  `myfixed['LepSoftEtaMaxVal']= '0.0'`: Maximal pseudorapidity of softer lepton.
- `myfixed['LepTriggerSpec']= '0.0'`:
  Apply trigger cut on leptons. 0: no cut; 1: apply cut.
  `myfixed['LepPtMin']= '0.0'`: Minimal transverse momentum of leptons in GeV.
  `myfixed['LepEtaMax']= '0.0'`: Maximal pseudorapidity of leptons.
- `myfixed['PhiLLCut']= '0.0'`:
  Cut on angle between charged leptons. 0: no cut; 1: apply cut.
  `myfixed['PhiLLMin']= '0.0'`: Minimal angle between charged leptons.
  `myfixed['PhiLLMax']= '0.0'`: Maximal angle between chaged leptons.
- `myfixed['PtMissCut']= '0.0'`:
  Cut on missing transverse momentum. 0: no cut; 1: apply cut.
  `myfixed['PtMissMin']= '0.0'`: Minimal missing transverse momentum in GeV.
  `myfixed['PtMissMax']= '0.0'`: Maximal missing transverse momentum in GeV.
- `myfixed['SpecMETCut']= '0.0'`:
  Apply special CDF missing E_{T} cut. 0: no cut; 1: apply cut. TO BE EXPLAINED!
  `myfixed['SpecMETMin']= '0.0'`: Minimal value for sum of missing transverse momenta.
- `myfixed['SumPtCut']= '0.0'`:
  Cut on sum of transverse momenta. 0: no cut; 1: apply cut.
  `myfixed['SumPtMin']= '0.0'`: Minimal value for sum of transverse momenta in GeV.
  `myfixed['SumPtMax']= '0.0'`: Maximal value for sum of transverse momenta in GeV.
- `myfixed['IsolationCut']= '0.0'`:
  Apply lepton isolation cut. 0: no cut; 1: apply cut.
  `myfixed['IsoConeSize']= '0.0'`: Cone size for lepton isolation cut.
  `myfixed['IsoEnergyFrac']= '0.0'`: Maximal fraction of hadronic energy in cone around lepton.
- `myfixed['InvMassCut']= '0.0'`:
  Cut on invariant mass of leptons. 0: no cut; 1: apply cut.
  `myfixed['InvMassMin']= '0.0'`: Minimal invariant mass of leptons.
  `myfixed['InvMassMax']= '0.0'`: Maximal invariant mass of leptons.
- `myfixed['PhotBasicCut']= '0.0'`:
  Apply basic photon cut: Cut on photon transverse momenta and photon pseudorapidity. 0: no cut; 1: apply cut.
  `myfixed['PhotPtMin1']= '0.0'`: Minimal transverse momentum of first photon.
  `myfixed['PhotPtMin2']= '0.0'`: Minimal transverse momentum of second photon.
  `myfixed['PhotEtaMax']= '0.0'`: Maximal photon pseudorapidity.
- `myfixed['PhotIsolCut']= '0.0'`:
  Photon isoliation cut. 0: no cut; 1: apply cut.
  `myfixed['PhotConeIso']= '0.0'`: Size of isolation cone around photons.
  `myfixed['PhotEtmax']= '0.0'`: Maximal hadronic energy in photon isolation cone in GeV.
- `myfixed['PtSoftCut']= '0.0'`:
  Cut on transverse momentum of softer photon. 0: no cut; 1: apply cut.
  `myfixed['PtSoftMin']= '0.0'`: Minimal transverse momentum of softer photon in GeV.

- `myfixed['PtmCut']= '0.0'`:
  Cut on average transverse momentum of photons. 0: no cut; 1: apply cut.
  `myfixed['PtmMin']= '0.0'`: Minimal average transverse momentum of photons in GeV.
  `myfixed['PtmMax']= '0.0'`: Maximal average transverse momentum of photons in GeV.
- `myfixed['PhotEtaHardCut']= '0.0'`:
  Cut on pseudorapidity of harder pohoton. 0: no cut; 1: apply cut.
  `myfixed['PhotEtaHardMin']= '0.0'`: Minimal pseudorapidity of harder pohoton.
  `myfixed['PhotEtaHardMax']= '0.0'`: Maximal pseudorapidity of harder pohoton.
- `myfixed['YstarCut']= '0.0'`:
  Cut on $Y^* = |\eta_1 - \eta_2|/2$ of photons. 0: no cut; 1: apply cut.
  `myfixed['YstarMin']= '0.0'`: Minimal Y^*.
  `myfixed['YstarMax']= '0.0'`: Maximal Y^*.

2.2 Collecting results

In the case of an exclusive computation the final result is obtained by combining the results of up to 99 different sectors. Again, we provide Python scripts and routines simplifying this task in the `scripts` subdirectory. But also in the case of inclusive calculations these scripts simplify the processing of results. Class definitions and routines are implemented in the file `CollectingClasses.py`, while simple user interfaces are provided with the files `CollectTot.py`, `Collectgg.py`, `CollectWW.py` and `CollectZZ.py`, respectively.

2.2.1 Inclusive calculation

- **Collecting results.** In order to collect the results from a single run, one has to provide the path to the result directory. Then an instance of class `InclusiveRes` must me created and its method `GetResults` called:

```
#-- Provide path to directory where the corresponding result file resides
thepath=os.path.join(thisdir,'results','Total','Total.mh.165_as0')

#-- Initialize a class instance of InclusiveRes in CollectingClasses.py
TheResults=CC.InclusiveRes(thepath)

#-- Collect results
TheResults.GetResult()
```

- **Print result file.** A result file (including the PDF error estimate if the run has been called with that option) can be written to an output file by providing the order of the computation and the name of the output file:

```
for porder in ['LO','NLO','NNLO']:
    resfile="Total.mh.165_as0_res%s.txt" %(porder)
    TheResults.Write(porder,resfile)
```

- **Saving and loading.** Once results have been read from the job's output file, the Python object can be saved. Thus, we can avoid to re-read the original output files. The Python object can be loaded in any other Python script for further processing by importing `CollectingClasses.py` and providing the file name used when saving.
  Saving:

```
savefile="Total.mh.165_as0.sav"
TheResults.Save(savefile)
```

  Loading:

```
module_dir = os.getcwd()
sys.path.insert(0,module_dir)
import CollectingClasses as CC

thepath=''
TheResults=CC.InclusiveRes(thepath)
savefile="Total.mh.165_as0.sav"
TheResults.Load(savefile)
```

- **Class methods.** The class `InclusiveRes` has the following methods apart from the above mentioned ones:
  - `InclusiveRes.Sig(korder)`: returns the cross-section at order `korder` (`'LO'`, `'NLO'` or `'NNLO'`).
  - `InclusiveRes.Err(korder)`: returns the integration error of the cross-section at order `korder` (`'LO'`, `'NLO'` or `'NNLO'`).
  - `InclusiveRes.PDFerror(korder)`: returns `[pdferror,pdferrorP,pdferrorM]` at order `korder` (`'LO'`, `'NLO'` or `'NNLO'`) where `pdferror` is the symmetric PDF error estimate while `pdferrorP` and `pdferrorM` are the assymmetric PDF error estimates following the MSTW08 prescription.

2.2.2 Exclusive calculation

- **Collecting results.** In order to collect the results from a single run, one has to provide the path to the result directory and the titles of distributions to be collected. Then an instance of class `InclusiveRes` must me created and its method `GetResults` called:

```
#-- Provide path to directory which contains the subdirectories for individual sectors
thepath=os.path.join(thisdir,'results','GG','GG.mh.120_as0')

#-- Define the histograms which you want to obtain
dists=['PT AVG','Ystar','PT LEAD','PT HIGGS','Y HIGGS']

#-- Initialize a class instance of InclusiveRes in CollectingClasses.py
TheResults=CC.ExclusiveRes(thepath,dists)

#-- Collect results
TheResults.GetResult()
```

- **Print result file.** A result file (including the PDF error estimate if the run has been called with that option) can be written to an output file by providing the order of the computation and the name of the output file. Further, the user can decide, which approximation should be used: $m_{\rm t} = \infty$ limit, top only or top and bottom contributions. The organization of the computation in suitable sectors allows to decide this at the collecting stage of the computation:

```
for porder in ['LO','NLO','NNLO']:
   for ctr in ['mtinfty','top','topbottom']:
        resfile="GG.mh.120_as0_res%s_%s.txt" %(ctr,porder)
        TheResults.Write(porder,ctr,resfile)
```

- **Redefine Wilson coefficients.** Redefining the Wilson coefficient can simply be performed without rerunning the computation. This is best done by creating a new Python object containing the original results and then call the routine `DefineWilsonCoeff`. Here, we add a second heavy quark with mass twice the top mass:

```
#-- First get a copy of the result object such that we do not have to read
#-- in result files again.
#-- Note: If we set TheResultsNew=TheResults, TheResultsNew and TheResults
#-- are the same object!
TheResultsNew=copy.deepcopy(TheResults)

#-- Now define C0, C1 and C2 (set to zero by default):
#-- Here we consider the example of an additional heavy quark with mass mq=2*mtop
muR=TheResultsNew.muR
mhiggs=TheResultsNew.mhiggs
mnewquark=2.0*TheResultsNew.mtop
Lr=math.log((muR/mhiggs)**2)
Lq=math.log((mhiggs/mnewquark)**2)
Nf=5e0
myC0=1e0
myC1=11e0/4e0
myC2=(2777e0/288e0+19e0/16e0*(Lr+Lq)+Nf*(-67e0/96e0 +1e0/3e0*(Lr+Lq)))
TheResultsNew.DefineWilsonCoeff(myC0,myC1,myC2)
```

The Wilson coefficient is defined in equation (1).

- **Saving and loading.** Saving and loading proceeds as in the inclusive case.
- **Class methods.** The class `InclusiveRes` has the following methods apart from the above mentioned ones:
  - `InclusiveRes.Sig(korder,ctr)`: returns the cross-section at order `korder` (`'LO'`, `'NLO'` or `'NNLO'`) and for approximation `ctr` (`'mtinfty'`, `'top'` or `'topbottom'`).
  - `InclusiveRes.Err(korder,ctr)`: returns the integration error of the cross-section at order `korder` (`'LO'`, `'NLO'` or `'NNLO'`) and for approximation `ctr` (`'mtinfty'`, `'top'` or `'topbottom'`).
  - `InclusiveRes.PDFerror(korder,ctr)`: returns `[pdferror,pdferrorP,pdferrorM]` at order `korder` (`'LO'`, `'NLO'` or `'NNLO'`) and for approximation `ctr` (`'mtinfty'`, `'top'` or `'topbottom'`) where `pdferror` is the symmetric PDF error estimate while `pdferrorP` and `pdferrorM` are the assymmetric PDF error estimates following the MSTW08 prescription.
  - `Hist(dist,korder,ctr)`: returns an array for the histogram with title `dist` at order `korder` (`'LO'`, `'NLO'` or `'NNLO'`) and for approximation `ctr` (`'mtinfty'`, `'top'` or `'topbottom'`).

3 Analysis routines

When running `FeHiPro` in the exclusive mode, the user has the option of defining histograms and add additional cuts. Example files are found in `FeHiPro/analysisSrc` (for plane Higgs studies), `FeHiPro/analysisSrcgg` (for di-photon decay channel), `FeHiPro/analysisSrcWW` (for WW decay channel) and `FeHiPro/analysisSrcZZ` (for ZZ decay channel). There are two files, which must be present in these directories:

- `initUserHist.F`
- `UserCuts.F`

In `initUserHist.F` Histograms are initialized. First, one has to specify the number of histograms with the variable `NUMHIST`. Then, for each histogram one has to provide a title (maximally 15 characters), the range and the number of bins, e.g.

```
HISTNAME(1)='PT HIGGS'
NUMBINS(1)=100
LOWEDGE(1)=0d0
HIGHEDGE(1)=200d0
```

One can specify up to 10 histograms. The integrand is not allowed to have more than 750 different components including the necessary components for PDF error sets if this option is turned on. In the main file a check is performed if the number of user defined histogram bins does not exceed the maximally allowed number of components.

The quantities to be filled into the histograms are specified in the `externalCuts.F`. The user can define arbitrary quantities that can be constructed from final state four momenta, including the Higgs momentum and vector boson momenta. The four momenta in laboratory frame are stored in an array

```
double precision part(11,4)
```

Particle j has four momentum components `part(j,1)` through `part(j,4)` where `part(j,1)` is the energy component, `part(j,4)` the component in the direction of the beam pipe and `part(j,2)` and `part(j,3)` are the components perpendicular to it. We list the particle assignements for the different decay modes in table 1. For example to fill histogram number 1 with the

	idecay					
j	0	1	2	3	4	5
1	Higgs	Higgs	Higgs	Higgs	Higgs	Higgs
2	None	W^+	W^+/Z_1	Z_1	Z_1	γ_1
3	None	ℓ^+	ℓ^+	ℓ^+	ℓ^+	None
4	None	ν_ℓ	ν_ℓ	ℓ^-	ℓ^-	None
5	None	W^-	W^-/Z_2	Z_2	Z_2	γ_2
6	None	ℓ^-	ℓ^-	ℓ^+	$\ell'^,+$	None
7	None	$\bar{\nu}_\ell$	$\bar{\nu}_\ell$	ℓ^-	$\ell'^,-$	None
8	Parton 1	Parton 1	Parton 1	Parton 1	Parton 1	Parton 1
9	Parton 2	Parton 2	Parton 2	Parton 2	Parton 2	Parton 2
10	Hard jet	Hard jet	Hard jet	Hard jet	Hard jet	Hard jet
11	Soft jet	Soft jet	Soft jet	Soft jet	Soft jet	Soft jet

Table 1: Particle assignment in `UserCuts.F`.

transverse momentum of the Higgs, the following lines have to be added in `UserCuts.F` before the filling and variables must be appropriately initialized.

```
pthiggs=dsqrt(part(1,2)**2+part(1,3)**2)
histval(1)=pthiggs
```

User defined cuts can also easily be specified by adding for example the following lines:

```
ptcut=50d0
pthiggs=dsqrt(part(1,2)**2+part(1,3)**2)
if (pthiggs.lt.ptcut) then
   UserCuts=0d0
endif
histval(1)=pthiggs*UserCuts
```

It is important both, to define `UserCuts=0d0` and to multiply the value for filling the histogram by `UserCuts`.

A Performance

B Running the code without Python scripts

The command for running `FeHiPro`assuming that the executable `fehip` as well as the input file `inputfile` are in the current directory is

```
> ./fehip -i inputfile -o resultfile -t /path2
```

where `/path2/pdfdat` is the path or a soft link to the PDF grid files. Relative paths are allowed for `path2`, e.g. `.` or `..` are valid. The output is written to `resultfile` prepended by the order of the calculation, i.e. `NLO.resultfile` for a NLO computation. Note that if one performs a NNLO computation, three output files will be created, one for each order LO, NLO, NNLO.

C Directory structure

The `FeHiPro` directory contains the following subdirectories:

- `analysisSrc`. General analysis files to be modified by the user.
- `analysisSrcgg`. Analysis files for the di-photon decay channel. To be modified by the user.
- `analysisSrcWW`. Analysis files for the WW decay channel. To be modified by the user.
- `analysisSrcZZ`. Analysis files for the ZZ decay channel. To be modified by the user.
- `mainSrc`. Main source files for initializing the computation and the constraint files for defining the jet function.
- `src`. Necessary source files of the original `FEHiP` code (computation in the $m_{\mathrm{t}} = \infty$ limit) [2].
- `hproSrc`. Source files for computing massive corrections derived from the NLO Monte-Carlo program `HPro`[3].
- `hggtotalSrc`. Source files for a fast inclusive computation at NNLO including mixed electroweak-QCD virtual contributions. Used in [4].
- `include`. Include files containing common blocks.
- `pvegas`. Source files for the non-parallel vegas algorithm as implemented by R. Kreckel, author of the parallelized vegas algorithm `pvegas` [5, 6].
- `Cuba-1.6_modified`. Contains Cuba library [1] as modified by us.
- `scripts`. Python script files for executing computations.

D Modified Cuba library

We provide a modified version of T. Hahn's Cuba library [1]. The added features are

- **Adaptation to a limited number of components.** The Cuba implementation of the vegas algorithm tries to adapt the grid such that required accuracy is reached for all components of the integrand. This can spoil convergence if the components have a rather different behavior.

  The modified vegas now allows to choose between adapting only to the first component (5^{th} bit of vegas argument `flags` set to 0) or to the first `ncadapt` number of components (5^{th} bit of vegas argument `flags` set to 1). `ncadapt` is an additional argument in the call to `vegas`.

- **Signal catching.** We have added signal catching. By pressing `Ctrl+C` the computation will halt after completion of the on-going iteration. In fact, the following signals are caught:
  - `SIGINT`: Computation will be halted after next iteration.
  - `SIGUSR2`, `SIGTERM`, `SIGXCPU`: Computation will be halted immediately.

  This feature allows to interrupt a computation and obtain results even if the required accuracy is not reached, yet. More importantly, on devices with limited runtime like a cluster the computation can be halted without loosing results.
- **Thermalization iterations.** Instead of variable `mineval` we have introduced variable `thermsteps`, i.e. now the user can define `thermsteps` number of thermalization iteration whose results will not be used for the final result. Accordingly, the χ^2 probability will be computed only from "real" measurements, i.e. total number of measurements minus the number of thermalization measurements.
- **Modified output.** We have modified the screen output.

References

[1] T. Hahn, Comput. Phys. Commun. **168** (2005) 78 [arXiv:hep-ph/0404043]. www.feynarts.de/cuba/

[2] C. Anastasiou, K. Melnikov and F. Petriello, Nucl. Phys. B **724** (2005) 197 [arXiv:hep-ph/0501130].

[3] C. Anastasiou, S. Bucherer and Z. Kunszt, JHEP **0910** (2009) 068 [arXiv:0907.2362 [hep-ph]].

[4] C. Anastasiou, R. Boughezal and F. Petriello, JHEP **0904** (2009) 003 [arXiv:0811.3458 [hep-ph]].

[5] R. Kreckel, arXiv:physics/9710028v1

[6] R. Kreckel, arXiv:physics/9812011v1