SensCalc: user guide

Maksym Ovchynnikov

E-mail: o.maxim@gmail.com

Contents

1 General overview and installation **1**
- 1.1 Why Mathematica? 2
- 1.2 Installation 2
- 1.3 Launching guidance 2
- 1.4 List of implemented FIPs 2
- 1.5 Performance 3

2 1. Acceptances.nb **3**
- 2.1 What users have to provide 5

3 2. FIP distribution.nb **7**
- 3.1 What users have to provide 7

4 3. FIP sensitivity.nb **8**
- 4.1 What users have to provide 9

5 4. Plots.nb **10**

Contents

1 General overview and installation

The code SensCalc consists of a few Mathematica notebooks that compute the number of events for various FIPs.[1] Four notebooks have to be run sequentially: Acceptances.nb, FIP distribution.nb, FIP sensitivity.nb, and Plots.nb, see Fig. 1.

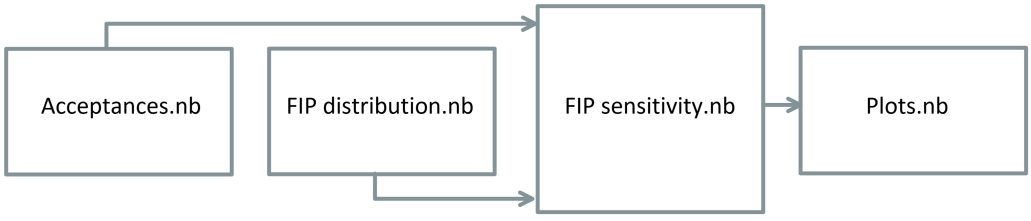

Figure 1. Description of the modular structure of SensCalc. The notebook Acceptances.nb produces the list of acceptances $\epsilon_{\rm az}$ and $\epsilon_{\rm dec}$ entering Eq. (2.1) of the accompanying paper for the selected experiment. The notebook FIP distribution.nb computes the distribution of FIPs $f(m, \theta, E)$ at the facility housing the experiment. The notebook FIP sensitivity.nb uses the input of the two previous notebooks to calculate the tabulated number of events, and calculates the sensitivity in the mass-coupling plane as a function of the remaining parameters, such as the minimal number of events and any additional model-specific parameters. Finally, Plots.nb produces the sensitivity plots from the output of the previous notebook.

[1] Contact the author: o.maxim@gmail.com

1.1 Why Mathematica?

Evaluating sensitivities of lifetime frontier experiments to FIPs is often much simpler than detailed event analysis or background evaluation. If computing the distributions of mother particles producing the FIPs externally, the sensitivity evaluation is split into relatively trivial computations such as calculating the FIP distribution function and evaluating the acceptances for the FIP and its decay products.

`Mathematica` is a great environment for such tasks. Namely, everything from computing the matrix elements of the FIP production and decay to working with the geometry of the experiment may be made inside it. In addition, `Mathematica` notebooks provide a convenient visual framework. Finally, there is a very friendly and experienced `Mathematica` community whose members may help with any question (and probably optimize the performance of `SensCalc` by another factor of ten).

`Mathematica` does not suit well for performing some low-level operations such as producing the phase space of many-body decays or propagating the particles through space. Fortunately, to improve this part, the `Mathematica` code may be simply compiled into a native machine code, which may improve the performance by a factor of a hundred and more.

1.2 Installation

The user needs to have installed `Wolfram Mathematica`, version 13.1 or higher. In addition, the user must install the `Mathematica` package `FeynCalc` and a `C` compiler recognized by `Mathematica`.[2] There are no other dependencies. The user can check whether the dependencies are properly installed by launching the notebook `0. Check for the needed soft.nb`.

1.3 Launching guidance

For the FIPs and experiments that are already implemented, the user just needs to launch the section *Just launch the code below to run the notebook* located at the top of each notebook. The relevant sections will then be launched automatically, and the notebook will prompt the user to specify the required inputs via dialog windows. If, however, the user wants to modify the model, geometry, or assumptions, they may change the code and inputs as described in the following sections.

1.4 List of implemented FIPs

The currently implemented FIPs are:

- The HNLs that are mixing with $\nu_e/\nu_\mu/\nu_\tau$
- Dark photons
- ALPs coupled to gluons, fermions, or photons
- Dark scalars
- Mediators coupled to the anomaly-free combinations of the baryon and lepton currents: $U_{\text{B-L}}(1)$, $U_{B-3L_\mu}(1)$, $U_{B-3L_e-L_\mu+L_\tau}(1)$, and $U_{B-L_e-3L_\mu+L_\tau}(1)$
- Millicharged particles (in public – only the production modes)

[2] For Windows machines, a suitable choice is `Visual studio community`, where the package "Desktop development with C++" should be installed.

1.5 Performance

Performance significantly depends on CPU and FIP. For most of the scenarios, the sensitivity may be computed from scratch within less than an hour. However, it may be much faster. For instance, Apple's new CPUs typically work much faster than Intel's CPUs. Also, if being launched once for an experiment belonging to the given facility, the output of the notebook `FIP distribution.nb` may be used for any other experiment from the same facility, which reduces timing substantially. Performance may be improved in the future.

2 1. Acceptances.nb

The output of the first notebook is data that includes:

1. A table with the following columns:

$$\{m, \theta, E, z, \epsilon_{\rm az}, \epsilon_{\rm dec}\} \tag{2.1}$$

2. The experiment- or facility-specific details: at which facility the experiment is located, whether the experiment has a calorimeter and a dipole magnet, what is the number of proton-proton or proton-target collisions, what are the production fractions of various SM particles such as mesons or heavy SM bosons, and the total branching ratio of the processes visible at the experiment.

The data is located in the folder
`Acceptances/`⟨`Given experiment`⟩
Depending on the specific FIP, experiment, OS, and CPU, the notebook typically runs for $\mathcal{O}(5)$ minutes per FIP. The runtime also significantly depends on the grid density and the number of simulated decays; the latter is parameterized by the number of ϕ values considered when evaluating the decay products acceptance (`NofPhiVals`), and by the number of decays simulated for each (m, E, θ, z) point (`ivaltest`).

The names of the implemented FIPs in this notebook are: `HNL-mixing-e`, `HNL-mixing-mu`, `HNL-mixing-tau`, `DP`, `ALP-gluon`, `ALP-fermion`, `ALP-photon`, `Scalar`, `B-L`, `B-3Lmu`, `B-3Le-Lmu+Ltau`, `B-Le-3Lmu+Ltau` (correspondingly to the list of the FIPs given in the section above).

Below is a description of how the notebook works, assuming that all the required ingredients are provided (the requirements are specified in the subsection below):

1. First, the user specifies the experiment for which they want to compute the acceptances, and selection criteria for the decay products (section *Specifying the experiment for which the acceptance should be computed*).

2. Next, the notebook computes the tabulated azimuthal acceptance $\epsilon_{\rm az}(\theta, z)$ (cf. *Computing the azimuthal acceptance*). It starts by evaluating the polar angle coverage of the decay volume and the detector, which is done automatically by discretizing the implemented geometry of the experiment. To ensure the correctness of the implementation of the experiment setup, the user should check the values of the parameters

   `ThetaDecVolGivenExperimentMin`,

   `ThetaDecVolGivenExperimentMax`,

   `ThetaDetGivenExperimentMin`,

   `ThetaDetGivenExperimentMax`,

and the visualization of the decay volume and detector in section *Visualization of the geometry*. Afterward, the notebook computes the grid of polar angles θ and z within the decay volume's coverage. Next, for each set of (θ, z), the list of values of the azimuthal angle ϕ that belongs to the decay volume acceptance is generated. Suppose the decay volume is not itself the detector. In that case, the notebook computes two grids of ϕ: one for when the FIP decays inside the decay volume and simultaneously points to the end of the detector (`EpsilonAzPhiListToDet`), and another one for when the FIP decays inside but does not point to it (`EpsilonAzPhiListNotToDet`). Once the computation of the azimuthal acceptance has finished, the user should check that the volume of the decay volume matches the value obtained using the computed acceptance $\epsilon_{\rm az}$ and Eq. (5) from the accompanying paper (cf. section *Grid with azimuthal acceptance*). Another cross-check is the visualization of the generated grid of the FIP's coordinates inside the decay volume, separately for the FIPs that point or do not point to the detector, and checking that the volume integral of the azimuthal acceptance matches with the total volume of the decay volume. These tests, together with displaying the E, θ, and z grids, are made in section *Grid with azimuthal acceptance and tests*.

2. To compute the decay products acceptance, one has first to simulate FIP decays at the rest frame of the decaying FIP. Consider a FIP `<FIP>` with mass `mFIP`. The phase space for the particular FIP decay channel `<process>` is generated on-flight by the block

   `PhaseSpaceFIPtoDecayProductsAtRest[<FIP>,<process>,mFIP,Nevents,ECALoption]`

   defined in the subsection *Phase space of decay products*. Here, `Nevents` is the number of decay events to generate, and `ECALoption` specifies whether to include neutral decay products such as photons or K_L^0 to the output. If some of the FIP's decay products are unstable, it repeats the routine `PhaseSpaceStableFromUnstableBlock`

   which decays it until only (meta)stable products are left: $\gamma, \pi^\pm, K^\pm, K_L^0, e^\pm, \mu^\pm, \nu$. For each decay product, the block returns the following columns: 4-momentum, mass, PDG id, electric charge, and stability. Decays into partons (such as FIP $\to u\bar{u}, c\bar{c}$, etc.) are treated by this routine as decays into a pair of the lightest charged mesons containing the given parton.

   Alternatively, if the given decay is into partons, instead of generating on-flight, it may be possible to use the phase space pre-generated in `MadGraph5+pythia8` for several characteristic masses `mFIP`. In this case, the partons are replaced with a bunch of hadrons appearing because of the showering and hadronization. The hadronized phase space is called by the block

   `PhaseSpaceFIPtoDecayProductsAtRestJets[<FIP>, <process>, mFIP]`

   The UFO files used to generate the phase space are located in the folder *UFO files*. The phase space is stored in the folder

   `simulated phase space`

   The names of the files must match the pattern

   `<FIP>_process_mass.txt`

   where `<FIP>`, `<process>` must exactly match the names of the FIP and the given decay process into the partons as defined in the notebook. An example of the filename is `Scalar_Jets-cc_3.8.txt`, which contains the phase space of the decay of the Higgs-like scalar with mass 3.8 GeV into hadrons resulting from the jets $c\bar{c}$.

   To account for various possible sets of the final states resulting from the hadronization, several sets of decay products exist per each decay channel. So the data has the structure `{{N_events,per channel, phase space},...}`, where `phase space` has the same structure

as the phase space simulated on-flight, while `N_events,per channel` is the number of events in the phase space, which serves as the relative weight of the given decay products set.

4. When computing the decay products acceptance (cf. *Decay acceptance calculation*), the notebook simulates the phase space of the relevant decay channels and calculates the decay acceptance for all possible pairs of decay products that have zero total electric charge. This is done by the block

   `FinalBlockMass[<FIP>, isim, HadronizationOption]`

   where `isim` is the number of decays simulated per mFIP, zFIP, thetaFIP, phiFIP, EFIP (the optimal value is around 1000), and `HadronizationOption` may be `True` of `False` depending on whether the user wants to hadronize partonic decays or not.

   The block first evaluates the purely geometric part of the decay acceptance by requiring the projection of the trajectory on the final plane of the detector to be within the detector cross-section. Then, if the decay products are within the geometric acceptance, it calculates the acceptance for the other cuts, such as the energy cut, etc. (see the subsection *Other cuts*). This is done with the help of the routine `DecayAcceptanceComp`. If a dipole magnet is placed somewhere, then "kicks" to the trajectory and the momentum components of the charged decay product in the direction transverse to the magnetic field are applied at the end of the magnet. The decay acceptances computed for the values of ϕ for which the FIP points (or does not point) to the end of the detector are then averaged, with each value being weighted by the corresponding azimuthal acceptance. Currently, `SensCalc` may generate on-flight 2-body, 3-body, and 4-body decays (many-body processes are, of course, possible when decay products are unstable). However, generic n-body decays may be implemented completely similarly to the 4-body decays.

   The user will be asked about the decay channels to be used to calculate the decay products acceptance.

2.1 What users have to provide

If the user wants to implement a new FIP, they need to provide the following:

1. The interpolated branching ratios of the decay processes

   `BrRatiosList[<FIP>, mFIP, <process>]`

   where `process` is the given decay process, `mFIP` is the FIP mass in GeV, and the associated list of the decay products decay channels `DecayTypeDecayProductsSetFIPs[<FIP>, process]`

   in the form {`product1, product2, product3, product4`}. If the processes are 2- or 3-body, the remaining products must be replaced with `"Null"`. The list of recognizable products is given in section *Properties of SM particles*. For instance, for a 3-body decay into $\pi^+\pi^-\gamma$, the list of decay products is {π^+,π^-,γ,`"Null"`}.

   For the implemented models, the branching ratios are stored in the folder

   `phenomenology/FIP/decay widths`

   and they are imported in the notebook in the section *Decays of various FIPs: processes, branching ratios, matrix elements*.

2. The squared matrix elements of the decay at the rest frame of the decaying particle. They are needed only for n-body decays with $n \geq 3$. Currently, `SensCalc` may only include it for 3-body decays; for decays with higher multiplicities, a unit matrix element is assumed. For 3-body decays, the squared matrix elements must depend on the decaying FIP mass `m`, the

energy of the 1st particle `E1`, the energy of the 3rd particle `E3`, and the decay products masses `m1,m2,m3`. The user may enter the matrix element and calculate its square using `FeynCalc`. The list of all squared matrix elements is given by

`Msquared3BodyFIP[<FIP>, <process>, E1, E3, m, m1, m2, m3]`

For the implemented FIP models, the matrix elements are defined in the section *Decays of various FIPs: processes, branching ratios, matrix elements*.

For the computation of the azimuthal and decay acceptances, the user has to provide:

1. The geometry and dimensions of the decay volume and detector, as well as the properties of the detector. For the detector equipment, the information that currently needs to be provided is the presence of the ECAL and of the dipole magnet, their positions, and the mean magnetic field of the magnet. This is done in the section *Geometry of different experiments and relevant cross-sections*.

   – If the decay volume and detector have a simple shape – box, cylinder, or annular cylinder, the only required inputs are the parameters describing the dimensions; they are listed in the subsection *Geometry* belonging to the subsection of the relevant experiment. An example of such parameters is `zToDecayVolumeExperiment[<Experiment>]`

   which specifies the longitudinal distance from the collision point to the beginning of the decay volume. If the detector has a more complicated shape, the user should implement its geometry in the section *Full geometry of the decay volume and detector*. Examples of already implemented non-standard geometries include the SHiP experiment and LHCb.[3] Regardless of how they were implemented, all the geometries are finally added to unified lists. For instance,

   `DecayVolumeGeometry[<experiment>]`

   contains the full geometry of the decay volume.

2. The integrated luminosity (for colliders) or the number of protons on target (for beam dump experiments) and the relevant production cross-section for the Standard Model particles (cf. subsection *Cross-sections*).

3. The relevant selection cuts on decay products. This is done via dialog windows (if the cuts are generic for, e.g., charged particles) or may be entered manually (if the cuts are different for various particles) in the section *Choosing the experiment and specifying its cuts*.

4. Conditions for the FIP to decay inside the decay volume and for the decay product to point/not point to the detector. These are

   `IfFIPinsideDecVol[zFIP, xFIP, yFIP, <experiment>]`

   `IfFIPtoDet[x1FIPproj, x2FIPproj, <experiment>]`

   `IfFIPnotToDet[x1FIPproj, x2FIPproj, <experiment>]`

   where `zFIP,xFIP,yFIP` are the coordinates of the decaying FIP (in metres), and `x1FIPproj`, `x2FIPproj` are the projections of particle's (FIP or its decay product) trajectory given its production/decay point and the presence/absence of the dipole magnet. `x1FIPproj` coincides with the x coordinate, while `x2FIPproj` may be the y coordinate (if the detector plane is transverse to the proton beam axis, as, e.g., for the SHiP experiment) or the z coordinate (in the other cases).

[3] Another example is the geometry of the ANUBIS experiment in the ceiling configuration; it is more complicated and hence stored in a different notebook `1. Acceptances. ANUBIS-ceiling`.

For the experiments with a simple shape of the decay volume and detector (e.g., box, cylinder, annular cylinder), these conditions may be calculated automatically by using the routines

`IfFIPinsideDecVolSimple[<experiment>]`

`IfParticlePointsToDetSimple[<experiment>]`

For more complicated cases, this has to be defined manually (see subsections *Conditions for the FIP to decay inside the decay volume and for the products to point to the end of the detector*).

3 2. FIP distribution.nb

The output of the second notebook is a tabulated distribution of the form

$$\{m, \theta, E, f^{(i)}\}, \tag{3.1}$$

where the last column is the value of the distribution function for the given (m, θ, E), and the production mechanism i. The distribution is normalized to one. The tables are located in the following folder:

`spectra/New physics particles spectra/⟨FIP⟩/⟨facility⟩/`

Depending on the specific FIP, OS, and CPU, the notebook runs for $\mathcal{O}(5)$ minutes (per production channel) when generating the distribution of a FIP that is produced by decays. For the production of ALPs in proton-target scatterings, $\mathcal{O}(30-40)$ minutes are currently required per target. This drop in performance will be addressed in a future version of the code. If the notebook has already been run once for a given facility, it does not need to be launched again for the other experiments housed at the same facility.

When launching the notebook, the user is prompted to choose the FIP and the facility at which the considered experiment is located. The notebook then computes the distributions of the FIPs produced by all implemented channels or by the channels selected by the user by evaluating the chapter *Computing distributions - ⟨FIP⟩*. This is done via tagging the sections needed for the given FIP production channels (the list of tags is summarized in `TagListProductionFIPs`).

For the FIPs produced in decays, the calculation is organized as follows: the notebook first samples random mother particles according to their distributions (see the routine

`BlockPointsFromSmoothDistribution`

from the section *Working with mother particles distributions*). For this, it imports the distribution of the mother particle from Sec. *Mother particles distributions* and accordingly generates `nsim` random mother 4-momenta.

Then (cf. section *Distribution of FIPs*), it generates the phase space of the FIPs in the rest frame of the decaying particles, boosts them (`Boosted⟨FIP⟩Block`), and finally produces a smooth FIP angle-energy distribution based on the generated data (`⟨FIP⟩distributionFrom⟨Mother⟩`). The distribution of FIPs produced by elastic scatterings is computed similarly.

The UFO files used to generate the distributions of FIPs from the DIS mechanism are located in the folder *UFO files*.

3.1 What users have to provide

The user has to provide:

1. The various production channels of the FIPs, in the form of branching ratios (if the same decaying particle can produce the FIP through multiple channels) and matrix elements (if the production channel is a 3-body decay, cf. section M^2 *of 3-body production for different FIPs*). The branching ratios for the implemented FIPs are located in the folder

`phenomenology/⟨FIP⟩/branching ratios/`

and the files follow the tabular schema

`FIP mass in GeV`, `Br ratio`

Users can add their branching ratios in any format, as long as they are imported appropriately.

2. The tabulated distributions of the mother particles in the form `polar angle in rad`, `Energy in GeV`, `Value of the distribution`. The distributions are located in the folder

   `spectra/SM particles/`

4 3. FIP sensitivity.nb

Once the user specifies the experiment and FIP (section *Specifying the experiment*), the notebook interpolates the tabulated functions obtained by the previous notebook and constructs the integral of the number of events (section *Number of events*). The simplest way is to obtain the number of events using the built-in Mathematica functions `Interpolation` and `NIntegrate`; this is the function `Nevents[m, coupling, ProdChannel]`, where `ProdChannel` is the given FIP's production channel. However, the resulting brute-force integrals are very slow ($\mathcal{O}$(1 second) per FIP's parameter space point "mass-coupling"). Given that the number of events has to be tabulated over a dense grid of FIP masses and couplings, the total computation time would be significant. In addition, if the integration domain in θ, E, z is much wider than the domain the events typically belong to, these integrals may return results with large errors. Instead, the subsection *Number of events - alternative* introduces a much faster and more stable way of calculating the number of events using quick grid mapping (the function `NeventsDiscret[m, ProdChannel, couplingslist]`).

The brute-force integral representation of the number of events serves to cross-check for the fast method (subsubsection *Number of events - alternative/Number of events/Test*). Typically, the agreement between the two methods is within 10%. Larger discrepancies may happen in a few cases:

1. Rarely, the θ, E, z grid density assumed by default for the computation of `NeventsDiscret`, `OutGrid`θ`final`, `OutGridEfinal`, `OutGridzfinal`, is insufficient. To exclude this reason, the user may just increase the density (say, changing `30` in

   `(10^Max[InGrid`$\theta\epsilon$`]-10^Min[InGrid`$\theta\epsilon$`])/30`

   by 100) to see whether the value of `NeventsDiscret` converges to `Nevents`.

2. A much more common reason is when the Monte-Carlo integration in `Nevents` fails to produce accurate results due to the abovementioned reasons.

The section *Acceptances and approximate lower/upper bounds computation* produces the acceptances that give a qualitative understanding of the contribution of the factors entering Eq. (1) of the accompanying paper (the FIP distribution function, the azimuthal acceptance, the decay probability, the decay products acceptance) on the sensitivity at the lower bound, where the number of events is

$$N_{\rm ev} = \frac{N_{\rm prod}}{c\tau_{\rm FIP}} \times \int d\theta dE dz f_{\rm FIP} \epsilon_{\rm az} c\tau_{\rm FIP} \frac{dP_{\rm decay}}{dz} \times \epsilon_{\rm dec} \tag{4.1}$$

where the integral is the FIP lifetime-independent. Namely, the data exported in the folder

`Auxillary data/⟨FIP⟩/⟨experiment⟩`

has the form

$\mathcal{I}_0$, $\mathcal{I}_1$, $\mathcal{I}_2$, $\mathcal{I}_3$

The integrals $\mathcal{I}_i$ are:

$$\mathcal{I}_0 = N_{\text{prod}}, \tag{4.2}$$

which shows the total number of FIPs produced at the given facility;

$$\mathcal{I}_1 = \frac{N_{\text{prod}}}{z_{\max} - z_{\min}} \int d\theta dE dz f_{\text{FIP}} \epsilon_{\text{az}}, \tag{4.3}$$

which shows the fraction of FIPs traveling in the direction of the decay volume times the length of the intersection of the FIP trajectory and the decay volume (normalized by the total number of the produced FIPs times the longitudinal length of the decay volume $z_{\max} - z_{\min}$);

$$\mathcal{I}_2 = N_{\text{prod}} \int d\theta dE dz f_{\text{FIP}} \epsilon_{\text{az}} c\tau_{\text{FIP}} \frac{dP_{\text{dec}}}{dz}, \tag{4.4}$$

which is the fraction of the FIPs decaying inside the decay volume;

$$\mathcal{I}_3 = N_{\text{prod}} \int d\theta dE dz f_{\text{FIP}} \epsilon_{\text{az}} c\tau_{\text{FIP}} \frac{dP_{\text{dec}}}{dz} \epsilon_{\text{dec}}, \tag{4.5}$$

which is the fraction of FIPs decaying inside the decay volume with the decay products satisfying the decay acceptance criteria. $\mathcal{I}_1$ shows the actual limitations on the sensitivity caused by the geometry of the experiments. $\mathcal{I}_2$ is proportional (up to $(c\tau)^{-1}$) to the number of decays assuming that $\epsilon_{\text{dec}} \equiv 1$. $\mathcal{I}_3$ is proportional to the total number of events at the lower bound. Basically, assuming that $N_{\text{ev}} \propto g^4$, where g is the FIP's coupling to the SM particles, the sensitivity at the lower bound is

$$g^2_{\text{lower}} = \sqrt{N_{\min} c\tau_{\text{FIP}}} / \sqrt{\mathcal{I}_4}\Big|_{\text{g = 1}}, \tag{4.6}$$

where $N_{\min}$ is the critical number of events determining the boundary of the sensitivity region. These lower bounds estimates are also performed in the notebook.

It may be possible to set the decay products acceptance to 1 when computing the number of events by changing the line

```
integrandtab = TableIntegrandDiscret[m, ProdChannel, "True"];
```

to

```
integrandtab = TableIntegrandDiscret[m, ProdChannel, "False"];
```

in `NeventsDiscret[m, ProdChannel, couplingslist]`.

The section *Exporting tabulated number of events* then exports the tabulated number of events in the form `FIP mass in GeV`, `FIP coupling`, `Number of events` for all the relevant production channels. The output is located in the folder

`Tabulated Nevents/`⟨`FIP`⟩`/`⟨`experiment`⟩`/`

Finally, the section *Computing sensitivities* imports the tabulated numbers of events, sums them over the production channels, and computes the sensitivity. The user must specify the minimum number of events and some model-specific parameters. The sensitivity curves can then be found in the folder

`Sensitivity domains/`⟨`FIP`⟩`/`⟨`experiment`⟩`/`

The typical running time is $\mathcal{O}(5-10)$ minutes.

4.1 What users have to provide

The user has to provide:

1. The probabilities to produce the FIP through various mechanisms. Typically, it is the branching ratio of the decay (for the decay process) or the scattering production probability per proton-proton collision (for the production by e.g. deep inelastic scattering). For already implemented FIPs, the tabulated probabilities are located in the folder

   `phenomenology/⟨FIP⟩/branching ratios/`

   They are imported and interpolated in the section *FIP phenomenology*, selected for the particular facility and convoluted with the production fractions of the mother particles (for the production via decays of secondaries) in sub-section *Importing distributions*.

2. The lifetime $c\tau$ (in meters) when the couplings of the FIP to SM particles are set to unit values (independently of their dimensionality), as well as the branching ratios of the various FIP decays. For the implemented FIPs, they are located in the folder

   `phenomenology/⟨FIP⟩/decay widths/`

   They are imported and interpolated in the section *FIP phenomenology*.

3. The tabulated FIP distribution functions and acceptances generated by the previous modules.

5 4. Plots.nb

This small notebook simply scans the folder containing the sensitivities for the chosen FIP, lists all the available curves that match the specified minimum number of events and model-specific parameters, imports the selected curves, and makes the sensitivity plot.