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.

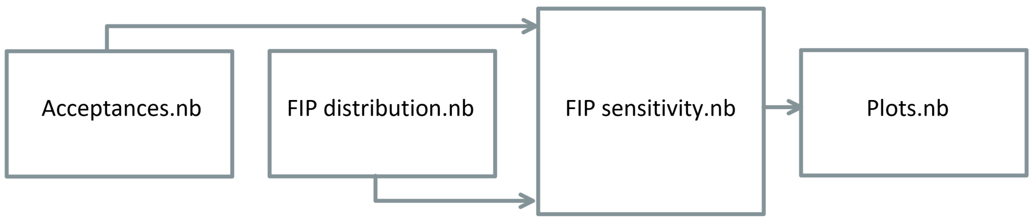

Figure 1. Description of the modular structure of `SensCalc`. The notebook `Acceptances.nb` produces the list of acceptances ϵ_{az} and ϵ_{dec} entering Eq. (2.1) of the accompanying paper for the selected experiment. The notebook `FIP distribution.nb` computes the distribution of FIPs $f(m, \theta, E)$ 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.

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`.

Why `Mathematica`?

`Mathematica` is a great environment for complex tasks like evaluating the sensitivities: assuming we have some specific input, it does not require a lot of external dependencies. 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. 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.

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 in 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]Contact the author: o.maxim@gmail.com

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

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
- $U_{\text{B-L}}(1)$ mediator

Performance

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

1. Acceptances.nb

The output of the first notebook is a data that includes:

1. A table with the following columns:
$$\{m, \theta, E, z, \epsilon_{\text{az}}, \epsilon_{\text{dec}}\} \tag{0.1}$$
2. The experiment- or facility- specific details: at which facility the experiment is located, whether the experiment has a calorimeter, 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.

The data is located in the folder
`Acceptances/⟨Given experiment⟩/Acceptance/`
Depending on the specific FIP, experiment, OS and CPU, the notebook runs for $\mathcal{O}(5-20)$ 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 (`isimulval`).

The names of the implemented FIPs in this notebook are: `HNL-mixing-e`, `HNL-mixing-`μ, `HNL-mixing-`τ, `DP`, `ALP-gluon`, `ALP-fermion`, `ALP-photon`, `Scalar`, `B-L` (correspondingly to the list of the FIPs given in the section above).

Below, there is the description of how the notebook works and requirements needed to use it:

1. Consider a FIP `FIP` with mass `mFIP`. The phase space for the particular FIP decay channel `SpecificProcess` is generated by the block

```
PhaseSpaceFIPtoDecayProductsAtRest[FIP, SpecificProcess,
                                        mFIP, Nevents, ECALoption]
```

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 in the output. If some of the FIP's decay products are unstable, it repeats the routine

`PhaseSpaceStableFromUnstableBlock`

which decays it until only stable products are left. For each decay product, the block returns the following columns: 4-momentum, mass, PDG id, electric charge, and stability.

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

1. The list of the decay channels `ProcessesList[FIP, ECALoption]`, where the second argument specifies whether one should include decays into purely neutral states. Currently, `SensCalc` supports 2-body, 3-body, and 4-body decays. However, generic n-body decays may be implemented completely similar to the 4-body decays.
2. The list of the decay products corresponding to the particular channel `SpecificDecay` from `ProcessesList`. For the given FIP `FIP`, it is specified by

   `DecayTypeDecayProductsSetFIPs[FIP, SpecificDecay]`
3. The list of the interpolated branching ratios. It is defined by `BrRatiosList[FIP, mFIP, SpecificDecay]`. The interpolations use tabulated branching ratios computed externally; for the implemented models, the latter are stored in the folder

   `phenomenology/FIP/decay widths`

   in the format

   `FIP mass in GeV`, `Br ratio`
4. The squared matrix element of the decay. It is 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, unit matrix element is assumed. The user must enter the matrix element for the process, and the squared matrix element is then calculated by `FeynCalc`. The list of all squared matrix elements is given by

   `Msquared3BodyFIP[FIP, SpecificDecay, E1, E3, m, m1, m2, m3]`

   where E_1, E_3 are energies of two of three decay products, m is the energy of the FIP, and m_i are masses of the decay products.

For the implemented FIP models, all these ingredients are included in the sections *Phenomenology of decays of various FIPs* and *Phase space of decay products at the rest frame of the decaying FIP.*

2. The azimuthal acceptance $\epsilon_{\rm az}(\theta, z)$ is computed in the following way (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 geometries. The user should check the values of the parameters

   `ThetaDecVolGivenExperimentMin`,

   `ThetaDecVolGivenExperimentMax`,

   `ThetaDetGivenExperimentMin`,

   `ThetaDetGivenExperimentMax`,

   as well as 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 that are within the coverage of the decay volume. 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 ϵ_{az} and Eq. (2.5) from the accompanying paper (cf. section *Grid with azimuthal acceptance*).

3. When computing the decay products acceptance (cf. *Decay acceptance calculation*), the notebook simulates the phase space of the relevant decay channels (cf. block `DecayAcceptanceBlock`), keeps or drops neutral particles depending on the presence or not of the calorimeter, and calculates the decay acceptance for all possible pairs of decay products that have zero total electric charge. It 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*). Depending on the presence of a dipole magnet, this is done with the help of the routines `DecayAcceptanceNoMagnetComp` or `DecayAcceptanceMagnetComp`; the second one introduces, at the end of the magnet, a kick to the trajectory and the momentum components of the FIP in the direction transverse to the magnetic field. 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.

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

   – 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. 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.

   – 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*).

   – The relevant selection cuts on decay products. This is done in the section *Choosing the experiment and specifying its cuts.*

2. FIP distribution.nb

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

$$\{m, \theta, E, f^{(i)}\}, \tag{0.2}$$

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 dark photons and ALPs coupled to gluons from the DIS mechanism are located in the folder *UFO files.*

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/`

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*) using the built-in Mathematica functions `Interpolation` and `NIntegrate`. However, these brute-force integrals are very slow ($\mathcal{O}(1$ second$)$). 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. Therefore, this representation of the number of events serves only as a cross-check that the method works. Instead, the subsection *Number of events - alternative* introduces a much faster but still accurate way of calculating the number of events using quick grid mapping. 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 has to specify the minimal 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.

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 (indendependently 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.

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.